

Sokratis K. Katsikas
Javier Lopez
Michael Backes
Stefanos Gritzalis
Bart Preneel (Eds.)

LNCS 4176

Information Security

9th International Conference, ISC 2006
Samos Island, Greece, August/September 2006
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Sokratis K. Katsikas Javier Lopez
Michael Backes Stefanos Gritzalis
Bart Preneel (Eds.)

Information Security

9th International Conference, ISC 2006
Samos Island, Greece, August 30 – September 2, 2006
Proceedings

Volume Editors

Sokratis K. Katsikas
University of the Aegean, Mytilene, Greece
E-mail: ska@aegean.gr

Javier Lopez
University of Malaga, Spain
E-mail: jlm@lcc.uma.es

Michael Backes
Saarland University, Saarbrücken, Germany
E-mail: backes@cs.uni-sb.de

Stefanos Gritzalis
University of the Aegean, Samos, Greece
E-mail: sgritz@aegean.gr

Bart Preneel
Katholieke Universiteit Leuven, Belgium
E-mail: bart.preneel@esat.kuleuven.be

Library of Congress Control Number: 2006931359

CR Subject Classification (1998): E.3, E.4, D.4.6, F.2.1, C.2, J.1, C.3, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-38341-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-38341-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springer.com

© Springer-Verlag Berlin Heidelberg 2006
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11836810 06/3142 5 4 3 2 1 0

Preface

This volume contains the papers presented at the 9th Information Security Conference (ISC 2006) held on Samos Island, Greece, during August 30 – September 2, 2006. The Conference was organized by the University of the Aegean, Greece.

ISC was first initiated as a workshop, ISW in Japan in 1997, ISW 1999 in Malaysia, ISW 2000 in Australia and then changed to the current name ISC when it was held in Spain in 2001 (ISC 2001). The latest conferences were held in Brazil (ISC 2002), UK (ISC 2003), USA (ISC 2004), and Singapore (ISC 2005).

ISC 2006 provided an international forum for sharing original research results and application experiences among specialists in fundamental and applied problems of information security.

In response to the Call for Papers, 188 papers were submitted. Each paper was reviewed by three members of the PC, on the basis of their significance, novelty, and technical quality. Of the papers submitted, 38 were selected for presentation, with an acceptance rate of 20%.

We would like to express our gratitude to the members of the Program Committee, as well as the external reviewers, for their constructive and insightful comments during the review process and discussion that followed. Moreover, we would like to thank all the members of the Organizing Committee for their continuous and valuable support. We also wish to express our thanks to Alfred Hofmann and his colleagues from Springer, for their co-operation and their excellent work during the publication process. Finally, we would like to thank all the people who submitted their papers to ISC 2006, including those whose submissions were not selected for publication, and all the delegates from around the world, who attended the ISC 2006 9th Information Security Conference. Without their support the conference would not have been possible.

August 2006

Sokratis K. Katsikas
Javier Lopez
Michael Backes
Stefanos Gritzalis
Bart Preneel

ISC 2006 9th Information Security Conference

General Co-chairs

Sokratis K. Katsikas University of the Aegean, Greece
Javier Lopez University of Malaga, Spain

Program Committee Co-chairs

Michael Backes Saarland University, Germany
Stefanos Gritzalis University of the Aegean, Greece
Bart Preneel Katholieke Universiteit Leuven, Belgium

Program Committee

A. Acquisti Carnegie Mellon University, USA
N. Asokan Nokia Research Center, Finland
V. Atluri Rutgers University, USA
T. Aura Microsoft Research, UK
F. Bao Institute for Infocomm Research, Singapore
J. Baras University of Maryland, USA
D. Basin ETH Zurich, Switzerland
G. Bella University of Catania, Italy
J. Benaloh Microsoft Research, USA
E. Bertino CERIAS, Purdue University, USA
A. Biryukov University of Luxembourg, Luxembourg
M. Burmester Florida State University, USA
S. De Capitani di Vimercati University of Milan, Italy
D. Catalano ENS, France
D. Chadwick University of Kent, UK
R. Cramer CWI and Leiden University, The Netherlands
B. Crispo Vrije Universiteit Amsterdam, The Netherlands
G. Danezis Katholieke Universiteit Leuven, Belgium
A. Datta Stanford University, USA
E. Dawson Queensland University of Technology, Australia
S. Furnell University of Plymouth, UK
V. Gligor University of Maryland, USA
D. Gollmann Hamburg University of Technology, Germany
H. Handschuh Spansion, France
D. Hofheinz CWI, The Netherlands
D. Hutter DFKI, Germany

J. Ioannidis	Columbia University, USA
A. Juels	RSA Laboratories, USA
T. Karygiannis	NIST, USA
S. Kokolakis	University of the Aegean, Greece
C. Lambrinouidakis	University of the Aegean, Greece
H. Lipmaa	Cybernetica AS & University of Tartu, Estonia
M. Mambo	University of Tsukuba, Japan
H. Mantel	RWTH Aachen, Germany
W. Mao	HP Labs, China
F. Massacci	University of Trento, Italy
M. Merabti	Liverpool John Moores University, UK
C. Mitchell	Royal Holloway, University of London, UK
A. Odlyzko	University of Minnesota, USA
E. Okamoto	University of Tsukuba, Japan
J. A. Onieva	University of Malaga, Spain
R. Oppliger	eSECURITY Technologies, Switzerland
G. Pernul	University of Regensburg, Germany
A. Pfitzmann	Dresden University of Technology, Germany
V. Rijmen	Graz University of Technology, Austria
P.Y.A. Ryan	University of Newcastle upon Tyne, UK
K. Sakurai	Kyushu University, Japan
P. Samarati	University of Milan, Italy
D. Serpanos	University of Patras, Greece
P. Tuyls	Philips Research and K.U. Leuven, The Netherlands and Belgium
J. Villar	Universitat Politecnica Catalunya, Spain
M. Yung	Columbia University and RSA Laboratories, USA
Y. Zheng	University of North Carolina at Charlotte, USA
J. Zhou	Institute for Infocomm Research, Singapore

External Reviewers

Asnar, W.	Clauí, Sebastian	Fernandez, Gerardo
Balopoulos, Theodoros	Cremonini, Marco	Fouque, Pierre-Alain
Batina, Lejla	Das, Tanmoy Kanti	Fuchs, L.
Bergmann, Mike	de Medeiros, Breno	Fukushima, Kazuhide
Bin Abd Razak, Shukor	De Win, Bart	Geneiatakis, Dimitris
Cardenas, Alvaro A.	Dobmeier, Wolfgang	Gonzalez, Juanma
Carvounas, Christophe	Doser, Juergen	Goubin, Louis
Cascella, Roberto	Fehr, Serge	Gouget, Aline
Chen, Haibo	Fergus, Paul	Gaujardo, Jorge

Gymnopoulos, Lazaros
Hankes Drielsma, Paul
Henricksen, Matt
Her, Yong-Sork
Herranz, Javier
Hilty, Manuel
Holmstroem, Ursula
Hori, Yoshiaki
Kambourakis, George
Karyda, Maria
Katzenbeisser, Stefan
Kiltz, Eike
Koeopf, Boris
Kolter, Jan
Koshutanski, Hristo
Krausser, Tina
Kunihiro, Noboru
Kopsell, Stefan
Lano, Joseph
Lee, Soo Bum
Lindqvist, Janne
Llewellyn-Jones, David
Meckl, Norbert S.
Muschall, B.
Naccache, David
Naliuka, Katerina
Neven, Gregory
Padr, Carles
Papadaki, Maria
Peng, Kun
Qiang, Weizhong
Schillinger, Rolf
Schlaeger, Christian
Schrijen, Geert-Jan
Seys, Stefaan
Skoric, Boris
Stefanidis, Kyriakos
Steinbrecher, Sandra
Sudbrock, Henning
Taban, Gelareh
Tu, Feng
Ueshige, Yoshifumi
van Le, Tri
Vercauteren, Frederik
Vigano, Luca
Volkamer, Melanie
Westfeld, Andreas
Wolf, Christopher
Woo, Chaw-Seng
Wu, Yongdong
Yatshukin, Artsiom
Yudistira, D.
Zannone, Nicola
Zhao, Yunlei
Zhong, Xiang
Zhou, Bo
Zhou, Juxiang
Zhu, Xusong

Table of Contents

Software Security

Extending .NET Security to Unmanaged Code.....	1
<i>Patrick Klinkoff, Christopher Kruegel, Engin Kirda, Giovanni Vigna</i>	

Transparent Run-Time Prevention of Format-String Attacks Via Dynamic Taint and Flexible Validation	17
<i>Zhiqiang Lin, Nai Xia, Guole Li, Bing Mao, Li Xie</i>	

Privacy and Anonymity

Low Latency Anonymity with Mix Rings.....	32
<i>Matthew Burnside, Angelos D. Keromytis</i>	

Breaking Four Mix-Related Schemes Based on Universal Re-encryption	46
<i>George Danezis</i>	

Weak k -Anonymity: A Low-Distortion Model for Protecting Privacy.....	60
<i>Maurizio Atzori</i>	

Protecting Data Privacy Through Hard-to-Reverse Negative Databases	72
<i>Fernando Esponda, Elena S. Ackley, Paul Helman, Haixia Jia, Stephanie Forrest</i>	

Block Ciphers and Hash Functions

Related-Key Rectangle Attack on 42-Round SHACAL-2	85
<i>Jiqiang Lu, Jongsung Kim, Nathan Keller, Orr Dunkelman</i>	

On the Collision Resistance of RIPEMD-160	101
<i>Florian Mendel, Norbert Pramstaller, Christian Rechberger, Vincent Rijmen</i>	

Digital Signatures

Blind Ring Signatures Secure Under the Chosen-Target-CDH Assumption	117
<i>Javier Herranz, Fabien Laguillaumie</i>	

Multi-party Concurrent Signatures	131
<i>Dongvu Tonien, Willy Susilo, Reihaneh Safavi-Naini</i>	

Formal Security Model of Multisignatures 146
*Yuichi Komano, Kazuo Ohta, Atsushi Shimbo,
 Shinichi Kawamura*

Cryptanalysis of Variants of UOV 161
Yuh-Hua Hu, Chun-Yen Chou, Lih-Chung Wang, Feipei Lai

Stream Ciphers

TRIVIUM: A Stream Cipher Construction Inspired by Block Cipher
 Design Principles 171
Christophe De Cannière

Cryptanalysis of the Bluetooth E_0 Cipher Using OBDD's 187
Yaniv Shaked, Avishai Wool

Encryption I

A Partial Key Exposure Attack on RSA Using
 a 2-Dimensional Lattice 203
Ellen Jochemsz, Benne de Weger

On the Integration of Public Key Data Encryption and Public Key
 Encryption with Keyword Search 217
Joonsang Baek, Reihaneh Safavi-Naini, Willy Susilo

Collusion-Free Policy-Based Encryption 233
Walid Bagga, Refik Molva

Pervasive Computing

Using Multiple Smart Cards for Signing Messages
 at Malicious Terminals 246
István Zsolt Berta

Diverging Keys in Wireless Sensor Networks 257
Michał Ren, Tanmoy Kanti Das, Jianying Zhou

Encryption II

A Generic Transformation from Symmetric to Asymmetric
 Broadcast Encryption 270
Ulrich Huber, Ahmad-Reza Sadeghi

Transparent Image Encryption Using Progressive JPEG 286
Thomas Stütz, Andreas Uhl

Network Security

Preserving TCP Connections Across Host Address Changes	299
<i>Vassilis Prevelakis, Sotiris Ioannidis</i>	
A Security Architecture for Protecting LAN Interactions	311
<i>André Zúquete, Hugo Marques</i>	
Simulation of Internet DDoS Attacks and Defense	327
<i>Igor Kottenko, Alexander Ulanov</i>	
SNOOZE: Toward a Stateful NetwOrk prOtocol fuzZER	343
<i>Greg Banks, Marco Cova, Viktoria Felmetzger, Kevin Almeroth, Richard Kemmerer, Giovanni Vigna</i>	

Watermarking and DRM

Rights Protection for Data Cubes	359
<i>Jie Guo, Yingjiu Li, Robert H. Deng, Kefei Chen</i>	
An Efficient Probabilistic Packet Marking Scheme (NOD-PPM)	373
<i>Huifang Yin, Jun Li</i>	

Intrusion Detection and Worms

Resistance Analysis to Intruders' Evasion of Detecting Intrusion	383
<i>Jianhua Yang, Yongzhong Zhang, Shou-Hsuan Stephen Huang</i>	
A Wireless Intrusion Detection System for Secure Clustering and Routing in Ad Hoc Networks	398
<i>Luciano Bononi, Carlo Tacconi</i>	
Anomaly Intrusion Detection Based on Clustering a Data Stream	415
<i>Sang-Hyun Oh, Jin-Suk Kang, Yung-Cheol Byun, Taikyeong T. Jeong, Won-Suk Lee</i>	
Robust Reactions to Potential Day-Zero Worms Through Cooperation and Validation	427
<i>K. Anagnostakis, S. Ioannidis, A.D. Keromytis, M.B. Greenwald</i>	

Key Exchange

An Authentication and Key Exchange Protocol for Secure Credential Services	443
<i>SeongHan Shin, Kazukuni Kobara, Hideki Imai</i>	
A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders	459
<i>Yvo Desmedt, Josef Pieprzyk, Ron Steinfeld, Huaxiong Wang</i>	

Security Protocols and Formal Methods

Formalising Receipt-Freeness	476
<i>H.L. Jonker, E.P. de Vink</i>	
Enhancing the Security and Efficiency of 3-D Secure.....	489
<i>Mohammed Assora, Ayoub Shirvani</i>	
Designing and Verifying Core Protocols for Location Privacy	502
<i>David von Oheimb, Jorge Cuellar</i>	

Information Systems Security

Delegation in a Distributed Healthcare Context: A Survey of Current Approaches.....	517
<i>Mila Katzarova, Andrew Simpson</i>	
Managing Information Systems Security: Critical Success Factors and Indicators to Measure Effectiveness.....	530
<i>Jose M Torres, Jose M Sarriegi, Javier Santos, Nicolás Serrano</i>	
Author Index	547

Extending .NET Security to Unmanaged Code

Patrick Klinkoff¹, Christopher Kruegel¹, Engin Kirda¹, and Giovanni Vigna²

¹Secure Systems Lab
Technical University Vienna
{pk, chris, ek}@seclab.tuwien.ac.at
²Department of Computer Science
University of California, Santa Barbara
vigna@cs.ucsb.edu

Abstract. The number of applications that are downloaded from the Internet and executed on-the-fly is increasing every day. Unfortunately, not all of these applications are benign, and, often, users are unsuspecting and unaware of the intentions of a program. To facilitate and secure this growing class of mobile code, Microsoft introduced the .NET framework, a new development and runtime environment where machine-independent byte-code is executed by a virtual machine. An important feature of this framework is that it allows access to native libraries to support legacy code or to directly invoke the Windows API. Such native code is called *unmanaged* (as opposed to *managed* code). Unfortunately, the execution of unmanaged native code is not restricted by the .NET security model, and, thus, provides the attacker with a mechanism to completely circumvent the framework's security mechanisms.

The approach described in this paper uses a sandboxing mechanism to prevent an attacker from executing malicious, unmanaged code that is not permitted by the security policy. Our sandbox is implemented as two security layers, one on top of the Windows API and one in the kernel. Also, managed and unmanaged parts of an application are automatically separated and executed in two different processes. This ensures that potentially unsafe code can neither issue system calls not permitted by the .NET security policy nor tamper with the memory of the .NET runtime. Our proof-of-concept implementation is transparent to applications and secures unmanaged code with a generally acceptable performance penalty. To the best of our knowledge, the presented architecture and implementation is the first solution to secure unmanaged code in .NET.

1 Introduction

With the growth of the Internet, applications are increasingly downloaded from remote sources, such as Web sites, and executed on-the-fly. Often, little or no knowledge exists about the author or her intentions. Therefore, users are susceptible to executing potentially malicious programs on their computers. Malicious programs contain code that executes in any unauthorized or undesirable way.

To secure users and increase the proliferation of mobile code, Microsoft recently introduced a new development and runtime framework called .NET [5]. This framework leverages the previous experiences gathered with the Java virtual machine concepts and includes a fine-grained security model that allows one to control the level of access associated with software built upon .NET. These applications are referred to as composed of *managed* code. The model significantly limits the damage that can be caused by malicious code. To address the important problem of backward compatibility and legacy code support, .NET also offers a mechanism to tie in native libraries. These libraries, however, execute outside of the .NET security model, and therefore are called *unmanaged code*. As a consequence, the usage of this feature in .NET applications may allow an attacker to completely circumvent the framework's security mechanisms, leading to the unrestricted execution of arbitrary code. This security problem is important because the use of unmanaged code will probably be common in future Windows .NET applications. Millions of lines of legacy native Windows code exist that will need to be integrated and supported over the next decade. Also, software engineering research [10] has shown that it is not realistic to expect existing applications to be entirely rewritten from scratch in order to take advantage of the features of a new language.

This paper describes our approach to extend the current .NET security model to native (unmanaged) code invoked from .NET. To this end, we use a sandboxing mechanism that is based on the analysis of Windows API and system call invocations to enforce the .NET security policy. Our approach ensures that all unmanaged code abides by the security permissions granted by the framework. Our primary contributions are as follows:

- Extension of existing sandboxing methods to .NET unmanaged code invocations.
- Two-step authorization of system calls by placing the security layer in the Windows API and the enforcement mechanisms in a loadable kernel driver.
- Separation of untrusted native library and trusted managed code into two separate processes by way of .NET remoting.

The paper is structured as follows. The next section provides an overview of the .NET framework and its security-relevant components. Section 3 introduces the design of our proposed system. Section 4 discusses the evaluation of the security and performance of the system and shows that our approach is viable. Section 5 presents related work. Finally, Section 6 outlines future work and concludes the paper.

2 Overview of the .NET Framework

Microsoft's .NET framework is an implementation of the Common Language Infrastructure (CLI) [6], which is the open, public specification of a runtime environment and its executable code. A part of the CLI specification describes the Common Type System (CTS), which defines how types are declared and

used in the runtime. An important property of the .NET framework is that it is type-safe. Type safety ensures that memory accesses are performed only in well-defined ways, and no operation will be applied to a variable of the wrong type. That is, any declared variable will always reference an object of either that type or a subtype of that type. In particular, type safety prevents a non-pointer from being dereferenced to access memory. Without type safety, a program could construct an integer value that corresponds to a target address, and then use it as a pointer to reference an arbitrary location in memory. In addition to type safety, .NET also provides memory safety, which ensures that a program cannot access memory outside of properly allocated objects. Languages such as C are neither type-safe nor memory-safe. Thus, arbitrary memory access and type casts are possible, potentially leading to security vulnerabilities such as buffer overflows.

The runtime environment can enforce a variety of security restrictions on the execution of a program by relying on type and memory safety. This makes it possible to run multiple .NET programs with different sets of permissions in the same process (on the same virtual machine). To specify security restrictions, the CLI defines a security model that is denoted as Code Access Security (CAS) [9]. CAS uses *evidence* provided by the program and security policies configured on the machine to generate permissions set associated with the application. Security relevant operations (for example, file access) create corresponding permission objects, which are tested with respect to the granted permission set. If the permission is not found in the granted set, the action is not permitted and a security exception is thrown. Otherwise, the operation continues.

Managed code executes under the control of the runtime, and, therefore, has access to its services (such as memory management, JIT compilation, or type and memory safety). In addition, the runtime can also execute *unmanaged code*, which has been compiled to run on a specific hardware platform and cannot directly utilize the runtime. In general, developers will prefer managed code to benefit from the services offered by the runtime. However, there are cases in which unmanaged code is needed. For example, the invocation of unmanaged code is necessary when there are external functions that are not written in .NET. Arguably, the most important library of unmanaged functions is the Windows API, which contains thousands of routines that provide access to most aspects of the Windows operating system.

To support interoperability with existing code written in languages such as C or C++ (e.g., the Windows API), the CLI uses a mechanism called *platform invoke service* (P/Invoke). This service allows for invocation of code residing in native libraries. Because code in native libraries can modify the security state of the user's environment, the .NET permission to call native code is equal to full trust [18]. Furthermore, native code launched by P/Invoke is run within the same process as the .NET CIL, and, as a consequence, malicious native code could modify the state of the .NET runtime itself. Microsoft suggests to only allow P/Invoke to be used to execute highly-trusted code. Unfortunately, users generally cannot determine the trust level of an application and will likely grant access also to non-trustworthy applications.

3 System Design

Our goal is to bring unmanaged native code invoked with P/Invoke from .NET *under the control* of the CAS rule-set. That is, we aim to combine the flexibility of unmanaged code with the security constraints enforced by managed code. When unmanaged code is executed, we assume that the attacker has complete control over the process' memory space and the instructions that are executed.

As a first approach, one could attempt to use the on-board operating system security model to enforce the desired .NET restrictions at the process level. That is, the downloaded application together with its native components is launched in a dedicated process. Then, operating system access control mechanisms are employed to restrict the privileges of this process such that the .NET Code Access Security settings are mirrored. Unfortunately, this is not easily possible. One problem is that Microsoft's Windows security model, though extensive, is different from the CAS model. That is, Windows security permissions differ from .NET permissions and do not provide a similar level of granularity. For example, in the CAS model, it is possible to allow a program to append to a file while simultaneously deny write access to the file's existing parts. In Microsoft Windows, on the other hand, the file system permissions have to be set to permit write access for a process to be able to append to a file. As another example, one can finely restrict network access to specific hosts using CAS, while this is not possible using OS-level Windows security mechanisms. Furthermore, Windows access control is based on user and role-based credentials. CAS, on the other hand, is based on the identity of the code, via its evidence. A comparable concept of evidence does not exist in the Windows security model. For example, it is not possible to define Windows security based on the URL the program was downloaded from.

Because the Microsoft Windows security mode is significantly different than CAS, we propose a dedicated security layer to extend the .NET code access security to unmanaged code. The goal of this security layer is to monitor the actions performed by the unmanaged code and enforce those restrictions specified by the CAS permission set. In the following sections, we discuss details about the design and implementation of our security layer.

3.1 Security Layer

The first design decision is concerned with the placement of the security layer. Ideally, this layer should be transparent to the application that it encapsulates. Also, it requires full access to all security-relevant functions invoked by the application (with parameters), so that sufficient information is available to make policy decisions. Finally, it must be impossible for malicious code to bypass the security layer.

The fundamental interface used by applications to interact with the environment, and the operating system in particular, is the Windows API. The Windows API is the name given by Microsoft to the core set of application

programming interfaces available in the Microsoft Windows operating systems. It is the term used for a large set of user mode libraries, designed for usage by C/C++ programs, that provide the most direct way to interact with a Windows system. We can, therefore, expect all security-relevant operations, such as file access, networking, memory, etc.¹, to pass through the Windows API.

In a first step, we decided to place the security layer between the Windows API and the native library. More precisely, we intercept calls to security-relevant Windows API functions and evaluate their function parameters. Fortunately, .NET security permissions map well to Windows API calls. Thus, we can evaluate the parameters of Windows API calls by creating and checking corresponding .NET permission objects. For example, we can evaluate the parameters of the `CreateFile`² API call and create a corresponding .NET permission object representing the filename and the requested action (*create* or *open*). Then, this permission object can be checked against the granted CAS permissions, appropriately permitting or denying the request.

To intercept Windows API functions, we make use of Detours [14]. Detours is a general purpose library provided by Microsoft for instrumenting x86 functions. This is achieved by overwriting the first few instructions of a target function with an unconditional jump to a self-provided function. Using this technique, we create hooks in security-relevant functions of the Windows API. The hook functions evaluate the parameters and create corresponding .NET permission objects. These permissions are then tested against the permission set granted to the application. If the requested action represented by the security permission is not permitted, a security exception is thrown. A valid request is passed on to the original Windows API call to perform the requested operation. By placing the security layer on top of the Windows API, it is possible to make the mechanism transparent to applications, and, in addition, it allows for comprehensive access to security-critical functions and their arguments.

Unfortunately, an attacker who has access to native code has great flexibility and can use a range of possible techniques to evade our naive security layer. The main reason is that the Windows API is user-level code that can be easily bypassed by interacting with the operating system directly. This could be achieved, for example, by invoking functions from `ntdll.dll`, which is the user space wrapper for kernel-level system calls, or by calling the system calls directly with assembly code. Another attack vector that needs to be mitigated is that parts of the .NET framework can be modified. Unmanaged code has complete and unrestricted access to the virtual address space that it is executed in. Unrestricted memory access can be leveraged by an attacker to overwrite management objects of the .NET runtime. For example, the variables holding the granted permission set could be modified. The attacker could also modify executable parts necessary for security enforcement, or simply tamper with objects on the managed heap, thereby crashing other .NET threads running on the same virtual machine. To protect from these kinds of attacks, the security layer has to

¹ For details on the Windows API, refer to [25].

² The name of this call is slightly misleading, as it is also used to open files.

shield the .NET runtime and concurrently executing processes from tampering with their allocated memory.

In the following Section 3.2, we introduce our approach to prevent unmanaged code from bypassing the Windows API when calling security-relevant functions. Then, in Section 3.3, we discuss our techniques to protect memory objects of the runtime from modifications.

3.2 Securing the Security Layer

In this section, we discuss our mechanism to prevent an attacker from bypassing the Windows API. To this end, we require a mechanism that allows us to enforce that certain user-mode library functions are called *before* corresponding operating system calls are performed. This mechanism is a second security layer that resides in kernel space. In a fashion similar to the previously mentioned layer at the API level, this second layer intercepts and analyzes operating system invocations. In particular, it enforces that each system call invocation must first pass through our security layer in the Windows API. To this end, the functions in the Windows API are modified such that subsequent system calls must be *authorized*. That is, whenever a security relevant Windows API function is invoked, this function authorizes the corresponding operating system calls that it is supposed to make. To make sure that only the security layer can authorize system calls (and not the native code controlled by the attacker), we have to ensure (i) that the authorization call originates from the security layer and (ii) that the security layer was not modified by the attacker. The mechanisms to enforce these two conditions are explained in more detailed later.

When unsafe code attempts to bypass the checks in the first security layer and performs a system call directly, the kernel space layer identifies this invocation as unauthorized and can abort the operation. The kernel driver is the only trusted component in the system, as it cannot be modified directly by a user process. Thus, if the attacker circumvents the Windows API, the invoked system call is not authorized and is therefore blocked by the driver.

Of course, the attacker could attempt to bypass the parameter evaluation in the security layer and jump directly to the instructions that grant the system call. We prevent this with a two-step authorization process. The check routine in the security layer immediately grants authorization for the system call. The parameters are then evaluated and, if any check fails, the security layer revokes its authorization. Thus, to authorize a system call, the attacker must always jump *before* the actual argument check routines and run through the entire process. Figure 1 shows the two-step authorization process.

The second security layer is implemented as a device driver loaded directly into kernel space. Russinovich [22] describes a method for hooking operating system calls in Windows NT by replacing the function pointer in the system call table. The driver employs this method to hook operating system calls and monitors the invocation of these calls from the unmanaged code. The security layer that resides at the Windows API level communicates with the kernel driver via IOCTL messages. These messages allow user space applications to communicate

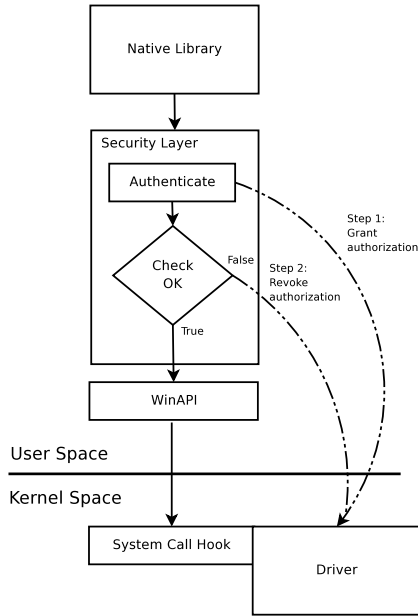


Fig. 1. Two-Step Authorization

with kernel-level drivers by passing buffers between them. In particular, `IOCTL` messages are used to perform authorization and revocation of system calls.

As discussed previously, the system must not allow the native code to communicate with the kernel driver directly (via `IOCTL` messages). Otherwise, the attacker could authorize (and later invoke) a certain system call without going through the security layer. Thus, only the security layer can be allowed to grant and revoke system calls. The problem is that both the security layer (at the Windows API level) and the native code are executed in the same address space, and it is not immediately obvious how a call from the security layer can be distinguished from one of the native code. To solve this problem, we permit `IOCTL` calls only from Windows API library code segments (where the security layer is implemented), and not from the native code itself (or from other segments such as the heap or stack). To this end, the system call handler for the `IOCTL` call first determines the address of the instruction that invoked the system call. If this address is not in the code segment of a library, it is not forwarded to the kernel driver. When the attacker attempts to jump directly to the instruction in the library that authorizes a call, the two-step authorization process ensures that arguments are checked properly. Otherwise, the authorization would be revoked immediately.

In addition, the correct operation of the two-step authorization process relies on the fact that the native code cannot alter the code executed by Windows API functions. Otherwise, it would be easy for an attacker to rewrite the code parts that check arguments or simply remove the statements that are responsible for revoking authorization when a CAS policy violation is detected. Fortunately,

ensuring that code sections are not modified is relatively straightforward. The reason is that executable code sections are stored in memory pages that are marked execute-only. Thus, to modify these sections, the attacker must first change the protection of the corresponding pages. To prevent this, the driver hooks the system call that handles page protection modifications. Pages containing executable code are typically marked as only `PAGE_EXECUTE`. This prevents reading or writing to any memory location in the page. To modify the functions, an attacker would have to change the page protection to allow for write access. To prevent this, we deny write modifications to any `PAGE_EXECUTE` pages. More precisely, we query the desired page protection before modification and do not allow elevation to write access for any page that has the execute flag set. This approach prevents an attacker from modifying executable code, but still allows for dynamic library loading. When a library is loaded dynamically, for example through the `LoadLibrary` call, memory is first allocated with `PAGE_READWRITE` protection [21]. After the library is loaded, the protection is changed to `PAGE_EXECUTE`. Because of this, the unmanaged code is effectively prevented from writing to executable pages in memory.

The security of the whole system relies on the fact that a user process cannot modify objects that reside in kernel space, and thus, cannot tamper with our second security layer. The astute reader might wonder why the security layer was placed in the Windows API in the first place, given the security advantages from placing it in kernel space. One important reason is the absence of a published documentation of the native API³, which is subject to changes without notice even between different service packs of Windows. In contrast, the Windows API is well-documented and explicitly designed to shield application code from subtle changes of the native API. In addition, Windows API calls exist that map to multiple system calls. In such cases, the Windows API function parameters indicate the actual purpose of the invocation and checks are easier to perform at the Windows API level than based on arguments of individual system calls.

As mentioned previously, unmanaged code cannot tamper with the driver because it is located in kernel space. We can, therefore, use the driver as a trusted storage for important data. In particular, to mitigate the danger of an attacker modifying the CAS permission set, we safely store it in the trusted storage. To this end, we serialize the permission set and store it in the driver before we launch any native code. This is, again, achieved with `IOCTL` messages. Note that permission sets are stored on a per-process basis. That is, multiple processes with different permission sets can be sandboxed at the same time. When checking a requested action, the security layer does not check against the (possibly modified) permission set residing in `.NET`. Instead, the security layer first retrieves the trusted permission set from the driver and then checks against this set. Of course, the permission set stored in the driver cannot be modified directly by the unmanaged code through another `IOCTL`, because that invocation would be trapped and checked with respect to the established permission set.

³ Even though [19] does an excellent job at documenting the native API, the documentation can never be complete without support from Microsoft.

In the previous discussion, the two steps of granting and revoking authorization were explained in the context of a process. However, when considering multi-threaded applications, this two-step authorization process would contain a race condition. This race condition can be exploited when one thread attempts a particular forbidden call, while another thread attempts to sneak in the same call between the time it is originally authorized and the time it is revoked. This problem is solved by granting and revoking authorization for system calls on a per-thread basis. That is, whenever the kernel driver is consulted to grant or revoke permissions for a system call, it checks the thread identified of the currently running thread instead of its process ID.

3.3 Remoting

Using security layers and the two-step authorization process, the CAS protection is successfully extended to unmanaged code. That is, the CAS model is enforced by monitoring all relevant interaction with the operating system and the permission set is safely stored in the trusted kernel driver. Unfortunately, the objects in the managed heap and data structures of the runtime can still be altered by an attacker, possibly causing the virtual machine or other .NET threads to crash or behave unexpectedly. Another problem is that there are system calls invoked by the runtime (or certain managed classes) that do not necessarily pass through the Windows API. Although these system calls are not authorized by our security layer, they are still valid. Of course, these calls must be permitted as blocking them would prevent managed classes from functioning correctly.

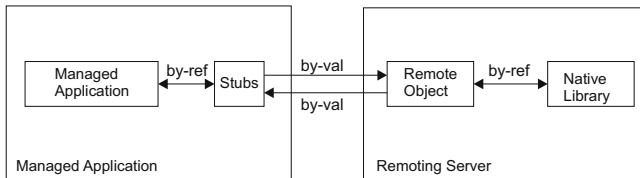


Fig. 2. Remote Parameter Passing

To protect the managed heap of .NET threads (and the runtime) and to make tracking of system calls easier, we isolate the unsafe code from the managed code that invokes it. More specifically, we create a process boundary between the managed code and the unmanaged code. Existing sandboxing techniques consider the entire process untrusted. For our purposes, however, we must distinguish between managed and unsafe code, even though these run in the same process. We therefore isolate the untrusted native library from the trusted managed code by running them in two different processes. In this way, we leverage the basic memory protection mechanisms offered by the operating system and prevent unmanaged code from accessing memory allocated by managed code.

When the native, unmanaged parts of an application are executed in a process different from the one where the managed part of the application resides, the question naturally arises how communication between these processes is realized. In particular, we need to explain how parameters and return values can be exchanged between the process that runs managed code and the process with the native code piece. While simple data types such as integers can be easily passed (copied) between address spaces, the situation is more difficult when complex data structures such as linked lists are involved. In these cases, the data structures have to be serialized by the sender and appropriately rebuilt by the receiver.

To accomplish the data exchange between the managed and the native processes, we make use of .NET remoting, the Remote Procedure Call (RPC) mechanism of .NET. To use .NET remoting, two proxy libraries have to be generated. The first proxy library contains the stubs for the native calls and is linked with the managed part of the application. More precisely, this library acts as an interceptor that replaces the original native library. It contains one method stub for each function of the unmanaged code that the managed code can invoke. Each method stub uses .NET remoting to invoke its corresponding method in the second proxy library. The second proxy library, called remote object, exposes a remote method for each function that the managed code uses in native libraries. These remote methods then perform the actual invocation of the native library in the remote process. Conceptually, the .NET remoting process can be viewed as an additional level of indirection between the managed code and the native libraries. Instead of passing values directly to the native code via the P/Invoke function, these values are first copied to the remote process using .NET remoting and only there passed to the native library. Note that both proxy libraries are automatically generated from the managed assembly. To this end, we use a combination of .NET reflection and an analysis of the disassembly of the intermediate language code. The goal is to obtain the required information to generate the proxy libraries, namely, the number of parameters of each native function and their respective types.

One problem that has not been discussed so far are parameters that are passed *by-reference* from managed code to the native library. The problem is that variables cannot be transferred across the process boundary with remoting when they are by-reference. This is because pointer values have no meaning outside the process address space. As a result, remoting parameters are always passed by-value. However, P/Invoke allows for by-reference parameters and we must take this into account. To solve this problem, we have to simulate by-reference parameter passing by copying the variables back and forth by-value. More precisely, the proxy library on the managed side transforms a by-reference argument into the corresponding value that is then copied to the remote process. Once the call into the native library is completed by the remote object, the stub method requests the parameter variables back. Then, the reference parameters are copied back into the original locations, as changes in the remote process must be reflected

in the original object. Figure 2 shows the process of simulating by-reference parameter passing.

The remoting server (see Figure 2) hosts both the remote object (which contains managed code) and the native library that should be confined. Before the unmanaged code is executed, the remoting server has to perform a number of initialization tasks. First, the Windows API hooks are installed to perform API function monitoring. Then, the .NET security manager is used to generate the granted permission set based on the evidence provided by the managed application. This permission set is then serialized to an XML format and sent to the trusted storage. Finally, the kernel driver has to be initialized. To this end, the remoting server registers its own process ID for subsequent monitoring. From that point onward, the remoting server process is subject to the CAS policy enforcement and can no longer perform any unauthorized system calls. Of course, the native library can freely tamper with the process memory and possibly crash the virtual machine or return arbitrary results to the managed code. However, such actions only affect this single process, while the managed code and the runtime (together with other threads) is successfully shielded by the process barrier. In particular, note that values returned by the unmanaged code are automatically integrated into the .NET type system when received by the proxy on the managed side. If values are returned that do not correspond to valid types, the situation is detected and an appropriate unmarshaling exception thrown.

4 Evaluation

To evaluate the proposed approach, we developed a proof-of-concept implementation of our system. Our prototype implements both the security layer at the kernel level and the layer at the Windows API level. Also, we support running the native process in a dedicated process with the automatic generation of the .NET proxy libraries. The system extends CAS to the following areas: file access, registry handling, and interaction with environment variables.

We investigated whether the current prototype achieves our stated goal of extending .NET's CAS mechanism to native libraries. We report on results of our simulations of the attack methods discussed previously. We continue by shedding light onto the performance penalty incurred by the design and conclude with experiments that demonstrate that our system can successfully isolate the native libraries of real-world applications.

4.1 Functionality

Functionality testing is directly linked to our stated goal. We would like to ensure that native code cannot perform actions that are restricted by the code access security (CAS) policy. For this purpose, we first constructed a CAS rule set that denies access to a certain file. The check of the file name to enforce this policy is performed in the `CreateFile` Windows API function, which in turn has to authorize the invocation of the corresponding operating system call. Then, we attempted to bypass our checks and illegitimately obtain access to this file.

In a first test, we attempted to bypass the Windows API function and called `NtCreateFile` from `ntdll.dll` directly. As expected, our kernel-level security layer denied the call as it was not authorized by the Windows API security layer. In the second approach, we decided to avoid using libraries altogether and used in-line assembly code to invoke system calls directly. Again, our kernel driver prevented the system call invocation. Next, we simulated an attacker's attempt to subvert the runtime or the security layer. We simulated this attack by attempting to modify an executable function. As expected, the driver hook for page protection denied this modification.

The results obtained from our attacks indicate that our system works as expected, and we successfully showed that all system call invocations must first pass through the security layer, and the checks therein.

4.2 Performance

After testing the system's functionality, we ran performance analysis to determine the overhead incurred by the security layer and, in particular, the remoting infrastructure. To this end, we conducted a series of micro benchmarks to measure the performance overhead of individual calls to native library functions. All experiments were run on a machine with an Intel Pentium 4 1.8GHz and 1GB of RAM, running Windows XP with Service Pack 2.

We anticipated the .NET remoting infrastructure to incur the largest performance penalty. To measure this penalty, we isolated the remoting infrastructure from the remaining system. For this, we modified our remoting server to not instantiate the security layer and to not interact with the driver. Our first test library function takes no parameters and returns no variables. The test function solely invokes the `CreateFile` function from `kernel32.dll` to create a file. The remoting server is hosted on the same machine, preventing network delays from skewing the results. The first entry (i.e., Test 1) in Table 1 compares the average running time over ten calls of a direct P/Invoke call to a call redirected over .NET remoting. As we expected, the .NET remoting mechanism creates a considerable performance penalty, which arises from the need to perform inter-process communication. In our next test, we used the remoting server as outlined in Section 3.3. That is, the security layer was in place and interacted with the driver. Our test function was the same as above, i.e., it took no parameters and returned no value. The second entry (i.e., Test 2) in Table 1 shows the average running time over ten calls. The results indicate that our security layer introduces no measurable performance penalty (less than one millisecond). Finally, we investigated how parameter passing affects performance. To this end, our next test compared the overhead produced by parameters in the .NET remoting call. This overhead stems from the need to marshal arguments at the sender and restore them at the receiver. The `CreateFile` call has seven parameters and one return parameter, which need to be serialized and exchanged between processes. The last entry (i.e., Test 3) in Table 1 shows that including parameters exacerbates the performance penalty.

Table 1. P/Invoke vs. Remoting

Test	Test Description	Direct Call (P/Invoke) (ms)	Remoting Call (ms)
1	No Security Layer	15	234
2	Active Security Layer	15	234
3	Active Security Layer + Function Parameters	15	286

While the overhead of a remote procedure call is an order of magnitude larger than invoking unmanaged code within a process, this is not surprising. Also, note that the in-process P/Invoke call incurs significantly more overhead than a regular function call. Thus, we do not expect this mechanism to be used frequently by performance-critical applications and believe that the increase in security clearly outweighs the performance loss.

4.3 Remoting

Another feature that we evaluated is the remoting infrastructure and the generated stub libraries. In particular, we want to ensure that we have not introduced limitations on parameter passing and that we maintain transparency for managed real-world applications that use native library components.

As mentioned in Section 2, an important reason for the introduction of P/Invoke and the ability to include unmanaged code into .NET applications is the need to call Windows API functions. Thus, we have to ensure that our protection infrastructure supports the invocation of (almost) all API functions. To test the ability of our system to call Windows API functions, we selected a representative subset of ten routines from important areas such as process management, file handling, and helper functions (all implemented in the `kernel32.dll`, the core Windows kernel library). We then tested whether these functions can be invoked from managed code running in a different process. We observed that our design successfully passed the relevant parameters across the process boundary via .NET remoting and invoked the native functions in the remote server process. After invoking the respective function, possible return parameters were successfully passed back to the original process.

Besides tests with Windows API functions, we also investigated our system when running real-world managed applications that make use of native library routines. To this end, we tested our infrastructure on two popular libraries: Sleepycat Software’s Berkeley Database [23] and the OpenGL graphics library [20].

Berkeley Database (BDB) is an embedded database. This is, the database engine component is compiled as a library and linked with the application. BDB is officially available as libraries for multiple languages such as C, C++, and Java. In addition, an unofficial C# wrapper [1] exists to port BDB to .NET. This wrapper uses P/Invoke to call the functions of the original BDB library. To test our system, we used the C# wrapper to invoke functions of the BDB library. More precisely, our test application uses the C# BDB wrapper to open a

database, store and retrieve records, and close the database. Function parameters include strings, integers and enums for supporting flags.

For testing OpenGL, we used a C# wrapper called CsGL [4] that encapsulates a native OpenGL library. To test our prototype, we exercised basic OpenGL functionality, such as filling the background of a window and drawing a rectangle. However, because most OpenGL functions use a similar syntax, we are confident that this covers the majority of OpenGL.

In both cases, our system automatically generated the necessary proxy libraries to split the managed part and the native library into two processes. That is, instead of invoking unmanaged library functions directly with P/Invoke, the parameters were first transferred to a remote process via .NET remoting. Only there were the native functions executed (via P/Invoke). Also, in case where a function returned a value, these values were properly returned to the managed application. This demonstrates that our system can automatically and transparently isolate native components from managed code.

5 Related Work

The system presented in this paper uses a *sandbox* to confine the execution of potentially untrusted applications. Sandboxing is a popular technique for creating confined execution environments that has been of interest to systems researchers for a long time.

An important class of sandboxing systems uses system call interposition to monitor operating system requests. That is, system calls are intercepted at the kernel interface. Then, these calls and their arguments are evaluated against security policies and denied when appropriate. Numerous approaches have been proposed [2, 11, 17, 3] that implement a variation of a sandboxing mechanism based on system calls. These approaches typically differ in the flexibility and ease-of-use of the policy language and the fraction of system calls that are covered.

One problem with kernel-level sandboxing mechanisms is the need to install the necessary policy enforcement infrastructure (e.g., kernel drivers or operating system modifications). To circumvent this problem, techniques [15, 12] have been proposed that rely on existing monitoring infrastructure in the kernel (e.g., APIs used for tracing and debugging such as *ptrace*) to intercept system calls, which are then processed by a monitor that resides in user space.

The main differences between our proposed approach and sandboxing techniques that operate on system calls are twofold. First, we do not only analyze the invoked system calls but can also force native code to go through user-mode libraries first (in our case, Windows API functions) before invoking a system call. That is, our two-step authorization process extends system call interposition to user libraries. The second difference is that we distinguish between a trusted, managed part and an untrusted, native part of an application, which originally run together in the same address space. To protect the managed code from malicious, unmanaged code, both parts have to be run in separated processes.

Forcing native code to go through user libraries can also be achieved with program shepherding [16], a method for monitoring control flow transfers during program execution to enforce security policies. The difference to our system is that program shepherding cannot prevent data values from being overwritten, a property that we obtain by executing managed and unmanaged code in two separate address spaces.

Being at the boundary between potentially untrusted user programs and the trusted kernel, system calls have received interest also from other areas of security research. In particular, system calls have been extensively used for performing host-based intrusion detection. To this end, specifications of permitted system calls were either learned by observing legitimate application runs [8] or extracted statically from the application [24, 7].

Finally, Herzog and Shahmehri [13] present an approach that extends the Java policy syntax for resource control. While we do not extend the .NET policy syntax *per se*, we extend its reach by applying it to native code.

6 Conclusions

The number of applications that are being downloaded from Web sites and automatically executed on-the-fly is increasing every day. Unfortunately, some of these applications are malicious. The .NET framework provides a security mechanism called Code Access Security (CAS) to help protect computer systems from malicious code, to allow code from unknown origins to run with protection, and to help prevent trusted code from intentionally or accidentally compromising security. CAS succeeds in restricting undesired actions of managed code. However, the permission to invoke unmanaged (i.e., native) code gives a potential attacker complete freedom to circumvent all restrictions.

This paper introduced a system to extend the CAS rule-set to unmanaged code. The evaluation of the proof-of-concept prototype of our proposed system shows that our design is viable. In particular, we successfully extended the CAS rule set to important Windows API functions. By confining a possible attacker to using the Windows API, we subjected unmanaged code to our security layer. Further, we successfully protected our system against possible attack vectors, such as circumvention of the security layer and memory corruption. To the best of our knowledge, the presented architecture and implementation is the *first solution* to secure unmanaged code in .NET.

References

- [1] Berkeley DB for .NET. <http://sourceforge.net/projects/libdb-dotnet>.
- [2] A. Berman, V. Bourassa, and E. Selberg. TRON: Process-specific file protection for the UNIX operating system. In *Winter USENIX Technical Conference*, 1995.
- [3] S. Chari and P. Cheng. BlueBox : A Policy-Driven, Host-Based Intrusion Detection System. In *Network and Distributed Systems Security Symposium (NDSS)*, 2002.

- [4] CsGL. <http://csgl.sourceforge.net/>.
- [5] .NET Framework Development Center. <http://msdn.microsoft.com/netframework/>.
- [6] ECMA. ECMA 335 - Common Language Infrastructure Partitions I to VI; 3rd Edition, 2005.
- [7] H. Feng, J. Giffin, Y. Huang, S. Jha, W. Lee, and B. Miller. Formalizing Sensitivity in Static Analysis for Intrusion Detection. In *IEEE Symposium on Security and Privacy*, 2004.
- [8] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A Sense of Self for Unix Processes. In *IEEE Symposium on Security and Privacy*, 1996.
- [9] A. Freeman and A. Jones. *Programming .NET Security*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003.
- [10] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall Inc., New York, 1991.
- [11] D. Ghormley, D. Petrou, S. Rodrigues, and T. Anderson. SLIC: An Extensibility System for Commodity Operating Systems. In *USENIX Technical Conference*, 1998.
- [12] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. In *6th USENIX Security Symposium*, 1996.
- [13] A. Herzog and N. Shahmehri. Using the Java Sandbox for Resource Control. In *7th Nordic Workshop on Secure IT Systems (NordSec)*, 2002.
- [14] G. Hunt and D. Brubacher. Detours: Binary Interception of Win32 Functions. In *3rd USENIX Windows NT Symposium*, 1999.
- [15] K. Jain and R. Sekar. User-level infrastructure for system call interposition: A platform for intrusion detection and confinement. In *Network and Distributed Systems Security Symposium (NDSS)*, 2000.
- [16] V. Kiriansky, D. Bruening, and S. Amarasinghe. Secure Execution Via Program Shepherding. In *11th USENIX Security Symposium*, 2002.
- [17] C. Ko, T. Fraser, L. Badger, and D. Kilpatrick. Detecting and Countering System Intrusions Using Software Wrappers. In *9th USENIX Security Symposium*, 2000.
- [18] .NET Framework Class Library Documentation - Security.Permissions. <http://msdn.microsoft.com/library/en-us/cpref/html/frlrfSystemSecurityP%ermissons.asp>, 2006.
- [19] G. Nebbett. *Windows NT/2000 Native API Reference*. New Riders Publishing, Thousand Oaks, CA, USA, 2000.
- [20] OpenGL. <http://www.opengl.org>.
- [21] R. Osterlund. Windows 2000 Loader, What Goes On Inside Windows 2000: Solving the Mysteries of the Loader. *MSDN Magazine*, March 2002.
- [22] M. Russinovich and B. Cogswell. Windows NT System-Call Hooking. *Dr. Dobb's Journal*, January 1997.
- [23] Sleepycat Software. Berkeley DB Database. <http://www.sleepycat.com/>.
- [24] D. Wagner and D. Dean. Intrusion Detection via Static Analysis. In *IEEE Symposium on Security and Privacy*, 2001.
- [25] Platform SDK: Windows API. <http://www.microsoft.com/msdownload/platformsdk/>.

Transparent Run-Time Prevention of Format-String Attacks Via Dynamic Taint and Flexible Validation

Zhiqiang Lin, Nai Xia, Guole Li, Bing Mao, and Li Xie

State Key Laboratory for Novel Software Technology
Nanjing University, 210093, Nanjing, China
{linzq, xianai}@dislab.nju.edu.cn

Abstract. Format-string attack is one of the few truly threats to software security. Many previous methods for addressing this problem rely on program source code analysis or special recompilation, and hence exhibit limitations when applied to protect the source code unavailable software. In this paper, we present a transparent run-time approach to the defense against format-string attacks via dynamic taint and flexible validation. By leveraging library interposition and ELF binary analysis, we taint all the untrusted user-supplied data as well as their propagations during program execution, and add a security validation layer to the `printf`-family functions in C Standard Library in order to enforce a flexible policy to detect the format string attack on the basis of whether the format string has been tainted and contains dangerous format specifiers. Compared with other existing methods, our approach offers several benefits. It does not require the knowledge of the application or any modification to the program source code, and can therefore also be used with legacy applications. Moreover, as shown in our experiment, it is highly effective against the most types of format-string attacks and incurs low performance overhead.

1 Introduction

Because of some innate features of C programming language such as lack of memory safety and function argument checking, ever since it became the mainstream programming languages of choice, there have been problems with the programs produced using it. Format string vulnerability, discovered about six years ago [1], is a case of such problems. It applies to all format-string functions in C Standard Library and still exists in many software (e.g., a recent last-3-months search in the NIST National Vulnerability Database had returned 28 matching records of this vulnerability [2]).

Due to the ability to write anything anywhere [3,4], attacks exploiting format-string vulnerabilities are extremely dangerous: they can lead to the program denial of service (e.g., to crash the program by using multiple instances of `%s`-directive), or can read sensitive data from nearly any readable location in the process memory (e.g., information leakage attack by using some `%x`-directives), or can write arbitrary integers to the area the attacker desires to with carefully crafting the format-string (e.g., the most dangerous `%n`-directive attack). Like buffer overflows, format string attacks are well-recognized as one of the few severe and truly threats to software security [5,15].

Many defensive methods against format-string attacks have been investigated in the past several years, including static checking (e.g., [22]), compiler extensions and runtime guarding (e.g., [15,16]), safe library functions (e.g., [13,14]), execution monitoring (e.g., [31]), and so forth. As discussed in Section 6 in details, these approaches are all valuable and they can catch real attacks. However, some of them such as FormatGuard [16], TypeQualifiers [17] and White-listing [15], require access to program source code for special analysis or recompilation, and hence exhibit limitations when applied to protect the source code unavailable software; and some of them although do not rely on program source code and appear almost transparent, they either tend to restrict code for the protection (e.g., preventing `%n`-directive in non-static format string [13]), or just provide small scope checking (e.g., only detecting malicious write of `%n`-directive associated with the return address or frame pointer [14]).

In this paper, we present an improved *dynamic* and *transparent* approach to the detection and prevention of format-string attacks. Specifically, we employ library interposition technique to intercept the `printf`-family functions in C Standard Library, i.e., `glibc` (we consider Linux platform in this paper), to do a security validation against the format string; intercept the I/O as well as string related functions to taint all the untrusted data originating from untrusted sources and their propagations during program execution. In order to get a good tradeoff between false positive and false negative, we provide two security policies, a default policy and a fine-grained policy. In our default policy, we validate the format string on the basis of whether it has been tainted and contains dangerous format specifiers. If so, a format-string attack is detected and we either signal an input/output error or kill the program to prevent it. With our fine-grained policy, we validate not only the tainted format string but also the untainted non-static one. For these untainted non-static, we check the `%n` corresponding argument whether or not points to those security critical areas such as return address, GOT (Global Offset Table) and DTORs (Destructor function table) [18]. We have developed a practical tool, FASTV (FormAt String Taint and Validation), to demonstrate this idea.

Compared with other existing methods, our approach offers several advantages:

- *Practical in application.* Our approach operates on a normally compiled binary program, and appears transparent to the protected software. This makes it practical to be used for a wide variety of programs, including proprietary and commodity programs whose source code is unavailable.
- *Effective against “real-world” attacks.* We collected 6 notable format-string attacks published at securityfocus.com [1,7,8,9,10,11]. Our approach successfully prevented all of them.
- *Easy to use.* For protections, users only need to set the environment variable and restart the program. Moreover, it can be simply set for the protection of either specific program or all processes on the machine.
- *Low run-time overheads.* As the experiment indicated, our approach only imposes about 10% run-time overhead on the protected software.

Our work makes the following contributions. We propose a novel dynamical taint and flexible validation approach to the detection and prevention of format-string attacks. In general, it is a *practical* and *efficient* solution in defending against these attacks.

Besides, we have implemented the prototype, and made an empirical evaluation showing the feasibility of our approach. In addition, perhaps more importantly, we believe our approach is also applicable for the prevention of other attacks such as SQL injection [6].

The rest of this paper is organized as follows. Section 2 presents a technical description of our approach. Section 3 describes the design and implementation of our prototype. Section 4 provides the experimental results on the effectiveness and performance evaluation. The security analysis and limitations are discussed in Section 5, followed by the related work in Section 6. Finally, Section 7 concludes the paper and outlines future work.

2 Approach Description

Since the root cause of format-string vulnerability lies in the format string, which is an argument to the `printf`-family functions, trusting the user-supplied input [3,4,14,15,16,17], the format-string argument (essentially, it is a string pointer) becomes our focus. If we can ensure the format-string argument is trustworthy or contains no dangerous format specifier when it is untrustworthy, we could hence prevent the format-string attacks. This is the key idea of our approach.

Upon the observation, we find out that the format-string argument passed to `printf`-family functions often falls into three categories:

- I. *Format-string argument pointing to a static string.*
- II. *Format-string argument pointing to a program dynamically generated string.*
- III. *Format-string argument pointing to a user-supplied input string.*

For the static string to act as a format-string argument, since it is constant and attackers cannot modify such strings (we do not cover static binary patching attack before program running in this paper), it is trustworthy and secure. We can distinguish it by ELF [18] binary analysis from the other two kinds, because static string resides in the program read-only data area and its address space is different from other program variables.

The second kind of format-string argument is internally maintained by the program itself, and in most cases is trustworthy. However, if attackers can influence the dynamically generated string (e.g., by buffer overflow attacks) or programmers carelessly deal with these data (e.g., forgetting to pass the corresponding argument), then it can also become untrustworthy. Therefore, we need to validate the format-string argument if tough checking required. Yet, we should mark this kind of format-string argument as trustworthy if no buffer overflow like attack occurs, because the bugs caused by carelessness should be eliminated before code ships.

The last kind of format-string argument is the common form of format-string vulnerability and undoubtedly the most dangerous. Our security validation mainly aims to find out this kind of format-string argument, which comes from user-supplied input and contains dangerous `%n`-directive (we currently focus on this specifier). It is an important part in our approach of how to identify and taint the untrusted user supplied data.

Here, suppose we have tainted all the untrusted data (its detail explanation is provided in the next section).

To detect the format-string attacks, we add a security validation layer to those `printf`-family functions in `glibc`. The security validation firstly determines whether the format string is static, since static format string is much more frequently used than that of other two kinds. (1) If the format string is static, we believe it trustworthy, and the function continues its original functionality (either to call the original function or execute code that is functionally equivalent). (2) Otherwise, the format string would be either dynamic generated or user-supplied (i.e., the tainted), and next we distinguish them based on whether it is tainted. (2.1) If the format string is tainted, then it is untrustworthy and we parse it to check whether it contains dangerous format specifiers. If it does, we report the format-string attack detected, and set the running `printf`-family function error; otherwise the function also continues its original functionality. (2.2) If the format string is untainted (this is the case for dynamically generated string), as stated it may be untrustworthy, and then if tough checking required we need to check it too. This is why we provide flexible policy. The default policy does not check it. In our fine-grained policy, the checking operation is to determine whether the `%n` corresponding argument points to return address, frame pointer, GOT (containing addresses of shared library functions) and DTORS (containing address of special destructor functions), because these areas are easily exploited as the attack target [4].

We employ ELF binary analysis and library interposition technique to achieve our goals. ELF binary analysis is used to identify the read-only static string and those attacker's target address, such as GOT and DTORS. Library interposition enables us to intercept those I/O and string propagation related functions in C Standard Library, so as to taint the user-supplied input. Another reason for using library interposition is that this technique does not require access to program source code, which makes our approach a more wider application.

3 Design and Implementation

We have developed a prototype, FASTV, to demonstrate our approach. The internal architecture of FASTV is illustrated in Figure 1. As shown in this figure, its core components are the dynamic taint of user-supplied input and flexible validation of format-string argument. We describe in detail these two parts design and implementation in this section, and additionally present the reactions when detected the format-string attacks. We provide the techniques to taint the user-supplied input in Section 3.1, and discuss how to validate the format-string argument according to different security policy in Section 3.2. The security reaction is provided in Section 3.3.

3.1 Dynamic Tainting Untrusted Data

Previous work [31] has suggested the use of taint analysis to track the input data that may lead to malicious attacks. However, their approach makes program run in an emulation environment and adds instrumentation to every data movement or arithmetic instruction, thereby imposing significant runtime overheads. In contrast, based on the

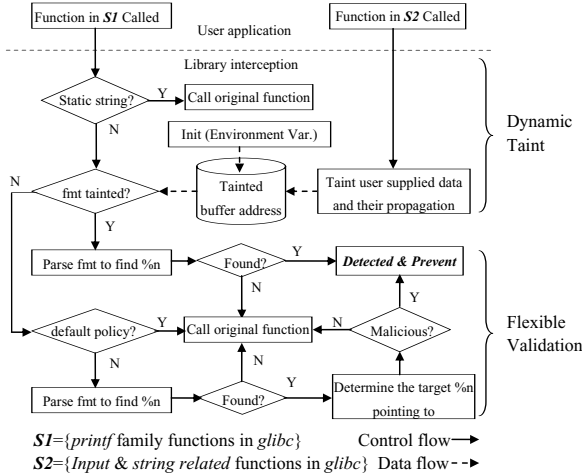


Fig. 1. Internal architecture of FASTV

observation that format string attack is usually caused by user-supplied input and these data are often string related, we could hence just intercept those *I/O* and *string* related functions in `glibc` instead of using hardware or software emulation based approach to track and taint the untrusted data.

Run-Time Representation of Tainted Data. In our approach, the taint operation is just to track the starting address and size information of those untrusted data in memory. We use a red-black tree [19] to store the tainted address and size in order to promote the performance. By associating taint operation with the starting address and size rather than other information, we can make our approach space cost little, and ensure the correctness of tainting in the presence of pointer aliasing. Why we do not store the content of every tainted data, the reason is when parsing the string content, we can get it from the tainted address.

Tainting Original Untrusted Data. For these directly user-supplied data, whenever the *I/O* related `glibc` functions are called, we intercept them and taint these input by inserting their buffer's starting address and size into our red-black tree. Here we should ensure that only one copy of a given buffer address exists in the tree. Thus, if the buffer has been tainted (by first searching the red-black to determine), we would not taint it again.

Note for the environment variable (e.g., user-supplied command-line data such as `argv`), we also need to taint them by inserting their associated address and size into the tree if tough checking required, because these data are also untrustworthy. In our current implementation, we have tainted these environment variables when loading FASTV by using the external variable `environ` to locate their address.

Tainting the Propagation of Original Untrusted Data. In our approach, tainting the propagation of original untrusted data is achieved by intercepting those *string* related `glibc` functions. Once these functions are called, we check firstly whether their corresponding original string (source string) is tainted. We determine this string is tainted on the basis of whether it exists in the red-black tree (its address equals to the node's address) or it belongs to the scope of certain node (we use the node's size to determine), for example, if address p is tainted and its length is 10, then " $p + 5$ " is also implicitly tainted since it lies in the scope of p .

After that, (1) if we find the source string has been tainted, then we taint the destination string. For the destination string, they may have been tainted previously, so we need to search the tree to determine whether it has been tainted. If it has not, we insert the destination starting address and size into our tree, and next the function continues. (2) If the source string is untainted, then we also need to ensure the consistency of the destination string untainted, and subsequently if it has been tainted before this time we need to delete it from or update the node's size in our red-black tree. We believe this consistency operation is reasonable since every string related operation is been guarded, and the most recent string to format functions would be the corresponding most recent modified.

3.2 Format-String Validation

The format-string validation is used to detect the format-string attacks. As described in Section 2, not every format-string argument is necessary for the validation, only those tainted data or when tough checking needed the dynamically generated string is included for. Thus, we provide a default policy and a fine-grained policy to handle the validation flexibly.

Default Policy. Our default policy is primarily to catch those user-supplied format string, which contains dangerous format specifiers. Specifically, if the tainted data contains `%n`-directive and is used as a format-string argument, then the format-string attack is detected.

The other remained two kinds of format string are not addressed in our default policy, and we regard them as trustworthy (this is reasonable as we have stated in Section 2). However, it might lead to false negatives if attackers gain control over the dynamically generated string. So in order to complement the default policy, we provide the other policy, fine-grained policy, to handle the dynamically generated format-string argument (no need to static string since it is secure).

Fine-grained Policy. The fine-grained policy aims to determine whether the `%n`-directive corresponding argument is secure when the format string dynamically generated. If this argument points to a program return address, frame pointer, entry of GOT or DTORS, then it is insecure and the attack is detected. Otherwise, the function continues.

We construct a reference table, which contains the base address and whole size information of program GOT and DTORS, when loading FASTV. We achieve ELF binary analysis to find out these security critical address as well as their associated size to create this table. As for the return address and frame pointer, these data are dynamically

changed and we cannot determine them by ELF binary analysis. Fortunately, Tsai et al. have proposed a solution to address this problem in Libsafe [14], and we adapt their approach here to locate the return address and frame pointer.

3.3 Reaction to Format-String Attacks

Many approaches when detected format-string attacks usually kill the running process. However, when attacks occur repeatedly, which is a common scenario with automated attacks, these protection mechanisms would lead to repeated restarts of the victim application and render its service unavailable. Thus, unlike their approaches we present a flexible mechanism for the preventions.

In our scheme, (1) if the format-string attacks are detected during the validation, then we report the format-string attack detected to `syslogd`, and set the `glibc` global variable `errno=EIO` to indicate this is an input/output error and let the program itself handle this problem. We believe for many server applications, they will take appropriate steps to deal with this input/output error. (2) If the output error is fatal (e.g., an intermediate output for after calculation) and the program itself ignores that, then here we also abort the running process in such a way as other approaches currently. We are planning to look for some alternative approaches (e.g., attack repair technique [20]) to remedy this (our aim is not to abort the running process).

4 Evaluation

We conducted a series of experiments to evaluate the effectiveness and performance of our approach. In order to compare our results with others, we chose Libsafe in that we both adapt library interposition, and White-listing, which is the most recent work, to do the evaluation. All the experiments were carried out on two 2.4G Pentium processors with 1G RAM running Linux kernel 2.6.3. The tested programs were compiled by `gcc` 3.2.3 with default option, and linked with `glibc` 2.3.2.

4.1 Vulnerability Prevention

In our evaluation, we focused on *real-world* format-string attacks and selected six of such programs. The vulnerability of these programs and our security test against them are described below.

- `wu-ftpd`. The `wu-ftpd` 2.6.0 and earlier is vulnerable to a serious remote attack in the “SITE EXEC” implementation, in which user-supplied data can be used as a format-string argument [1]. For the security test of this program, we exploited the return address as the `%n` target.
- `rpc.statd`. The `rpc.statd` program (for the version of `nfs-utils` before 0.1.9.1), passes user-supplied data to the `syslog` function as a format string. Thus, attackers can inject malicious code to be executed with the privileges of the `rpc.statd` process [7]. In our test, we tried to overwrite GOT entries as the `%n` corresponding pointer.

- `splitvt`. The `splitvt` before 1.6.5 program exists a vulnerability in the command line with `-rcfile`, which is not properly handled as a format-string argument [8]. Our attack test was targeted return address.
- `rwhoisd`. The `rwhoisd` 1.5 server contains a remotely exploitable format-string vulnerability, which allows attackers to execute arbitrary code on affected hosts by supplying malicious format specifiers as the argument to the `-soa` directive [9]. Again, our attack test was to patch the return address.
- `pfinger`. A format-string vulnerability exists in `pfinger` 0.7.5 through 0.7.7, which allows remote attackers to execute arbitrary command via format-string specifiers in a `.plan` file [10]. For this program test, we also tried to overwrite the return address.
- `tcpflow`. The `tcpflow` 0.2.0 program contains an exploitable format-string vulnerability during the opening of a device with the command-line argument. Thus, local users can gain an unauthorized root shell by this vulnerability [11]. We made our security attack target on GOT entries to test this program.

The results of our effectiveness evaluation are presented in Table 1. As shown in this table, our approach, not only the default policy (DP) but also the fine-grained policy (FP), successfully prevented all the format-string attacks listed above. This is expected because all the security test was used the user-supplied data which comes from local or network to launch the format-string attack.

For the White-listing approach, though it also reliably fixed all the vulnerabilities, it may lead to denial of service attack for some cases (though in our protections `pfinger` and `tcp-flow` aborted the running process, we should note this is the behavior of program itself). The Libsafe approach, also as expected, missed the attack which does not target return address or frame pointer, and only caught 4 out of the 6 attacks.

Table 1. Results of effectiveness evaluation

CVE#	Program	Libsafe	White-listing	FASTV(DP)	FASTV(FP)
CVE-2000-0573	<code>wu-ftpd</code>	D & A	D & A	P & C	P & C
CVE-2000-0666	<code>rpc.statd</code>	M	D & A	P & C	P & C
CVE-2001-0111	<code>splitvt</code>	D & A	D & A	P & C	P & C
CVE-2001-0913	<code>rwhoisd</code>	D & A	D & A	P & C	P & C
CVE-2001-1215	<code>pfinger</code>	D & A	D & A	P & A	P & A
CAN-2003-0671	<code>tcp-flow</code>	M	D & A	P & A	P & A

D: Detected, A: Aborted, P: Prevented, C: Continued, M: Missed

4.2 Performance Overhead

In order to test the performance overhead of our approach, we first did the micro-benchmark test to measure the overhead at the function call level, and then measured the overall performance at the application level by running a typical `printf`-intensive

application `man2html` (to test the overhead of *print* and *string* related functions), and a network program `tcpdump` (to test the overhead of *input* related functions, such as `read`, `recv`). All the tested programs were run multiple times with the highest priority in single-user-mode except for `tcpdump` which run in the network-mode due to its network requirement.

Micro Benchmarks. To determine the overhead of per `printf`-style function, we ran a serial of simple programs consisting of a single loop containing one single `sprintf` call, with a varied number of format string length. We choose `sprintf` in that we can use this function to test the performance of both format-string validation (parse and check the format string) and related string taint (taint the relevant destination string if the corresponding printed string is tainted), and its performance is greater than that of other `printf`-family functions. In addition, we choose `strcpy` to test the micro-benchmark of string related functions since in our approach we also wrapper them.

With static format string which contains no format specifiers, our approach as well as Libsafe added almost no performance overhead (the performance added rate is zero). As for White-listing, it had a different performance added rate, which is greater than ours and Libsafe's. To be more specific, when the format-string length is not too long in our test, e.g., less than 100, White-listing only incurred little performance overhead; and when format-string length is added, e.g., to 1k, it added the performance overhead of $3\mu\text{s}$ (about 75%).

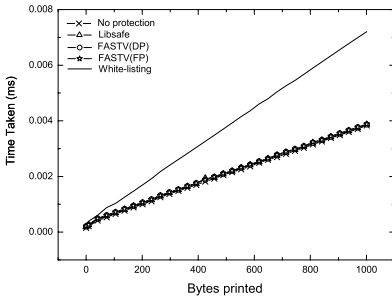
For dynamically generated string which contains two `%n` directives, our approach with DP did not add any overhead, which is similar to Libsafe. This is expected, because our default policy does not check these program dynamically generated string. As for FP of our approach, which will check all the dynamically generated string, the performance added rate was less than White-listing's; and in the worst case of our test, it added $2.4\mu\text{s}$ (about 60%), and White-listing added $3.5\mu\text{s}$ (about 90%).

With user-supplied different length format string (it is just a performance test here, though it appears insecure) which contains no format specifier, our overhead for both DP and FP was similar to the result of dynamically generated string with FP: the performance added rate was less than White-listing's.

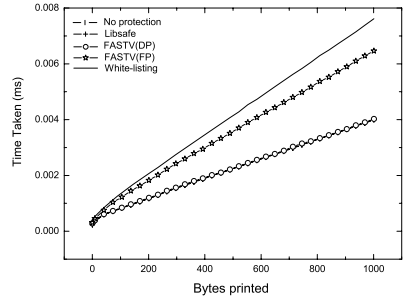
We also tested `vsprintf` by replacing the printing loop with `vsprintf`, and correspondingly modifying its relevant argument, to test the performance of per `vprintf`-style function call. We observed a similar performance overhead to `sprintf` function for the above three test cases, respectively.

For the micro-benchmark of `strcpy` function, in our test, except for Libsafe which improves the performance by replacing `strcpy` with `memcpy` [14] (this is based on the fact that copying with `memcpy` is 6 to 8 times faster than that of `strcpy` for large buffers [21]), our approach as well as White-listing almost did not add any performance overhead.

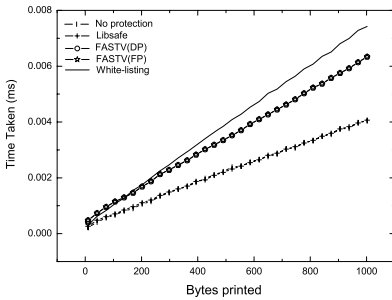
All the results for our micro benchmarks are depicted in Figures from 2(a) to 2(d). Some of these overheads may seem relatively high, but we stress that these are micro benchmarks and not realistic programs. And as we show below for real-world applications, our approach only incurs a little performance overhead.



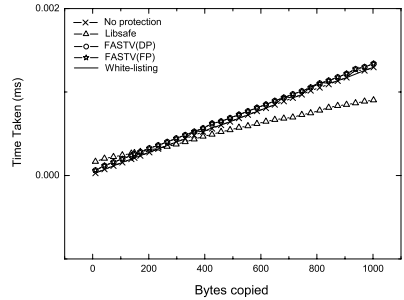
(a) Sprintf micro-benchmark with static format string containing no specifiers



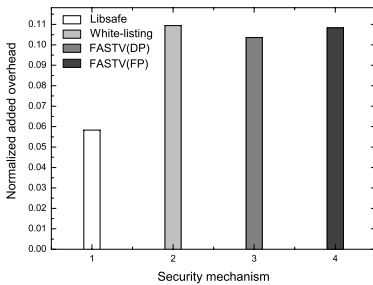
(b) Sprintf micro-benchmark with dynamically generated format string containing 2 %n directives



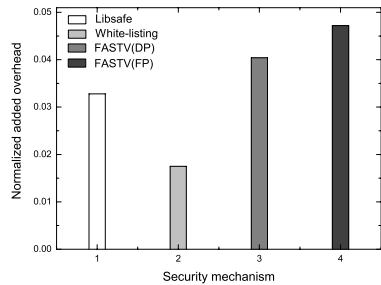
(c) Sprintf micro-benchmark with user-supplied format string



(d) Strcpy micro-benchmark with user-supplied string



(e) Man2html macro-benchmark



(f) Tcpdump macro-benchmark

Fig. 2. Results of benchmark test

Macro Benchmarks. We used `man2html` program to evaluate the macro-benchmark. Our test was to batch translate 1k man pages which is 129.6k bytes each, via `man2html-1.6`. The result for the macro benchmark of this program is presented in

Figure 2(e). As shown in the Figure, our approach incurred the performance overhead of 10.36% with DP and 10.84% with FP, which is a little less than that of White-listing with the overhead of 10.94%. For the Libsafe, the performance overhead was 5.83%.

We also tested our approach against a network program `tcpdump-3.9.4`. We ran this program by capturing 1k network packets in a high-speed transferring LAN. Our approach added the overhead of 4.04% with DP and 4.72% with FP. Libsafe added 3.28%, and White-listing added 1.75%. Figure 2(f) summarizes the result of this program.

5 Discussion

In this section, we discuss the false positive and false negative of our approach with different security policy, and its limitations when applied to the software protections.

False Positive and False Negative. As stated, our default policy guards all the dangerous tainted data partly or completely acting as the format string, there would be few false negatives (false negatives may exist in the un-caught string propagation if the application does not use the string related `glibc` functions) in this policy when handling user-supplied data. From theoretic analysis, we could find our default policy may have false positives, but from practice we could say almost *no* false positive. We have examined a large number of real-world applications and found none of them needs user-supplied `%n`-directive except for attacks. We believe this case, i.e., requiring users to input the `%n`-directive, does not exist in normal released software. So we believe it is not a common case and does not deserve our attention.

While our default policy is adequate for the format-string attack prevention in most circumstances, those untainted dynamically generated string might cause false negatives since we ignore the validation of these untainted data. As a result, we provide the fine-grained policy to complement our default policy. In our fine-grained policy, there are very few false positives since we used the mis-use detection approach (each `%n` corresponding argument is validated whether or not pointing to our guarded area). But there are false negatives in this policy, which is caused by the limitations described below.

Limitations. One of the limitations of our approach is in the fine-grained policy, we only guard the most common attacked areas: return address, frame pointer, GOT and DTORS. If an attack makes use of a program legal function-pointer for the `%n` corresponding written address, we would not be able to detect it. Because there is no useful information (e.g., type to tell us this is a function pointer) in the normal binary file to let us identify program legal function-pointers from other variables. As for the GOT and DTORS, due to a different memory region they reside in, we can find out their address via ELF binary analysis.

There is another limitation. Our approach requires the program being dynamically linked (this is because library interposition technique only intercepts the function references to dynamic library). However, the statically linked applications are not too much used if we consider Xiao's study that 99.78% applications in Unix platform are dynamically linked [12]. In addition, if the program invokes its own functions instead of `glibc` for *I/O* and *string* manipulation, our approach also would not work.

6 Related Work

A considerable amount of approaches have been developed for the defense against format-string attacks. Those related to ours could be divided into three categories: compile-time analysis, run-time techniques, and combined compile-time/run-time techniques.

Compile-Time Analysis. This technique typically analyzes and/or instruments program source code to detect possible format-string attacks. PScan [22] is such a kind of simple and notable tool for discovering format-string vulnerabilities. It works by looking for the common case of the `printf`-style function in which the last parameter is the format string and none-static. Similar to the functionality of PScan, `gcc` itself also provides a compiler flag, “-Wformat=2”, to cause `gcc` to complain about the non-static format string [23]. Both PScan and the “-Wformat” enhancement operate on the lexical level, and they offer the advantage of fixing bugs before software release. However, these two approaches are less complete, and usually subject to both missing format-string vulnerabilities and issuing warnings about safe code.

Compile-time analysis with taint technique is useful to find out bugs or identify potentially sensitive data. Perl’s taint mode [24] is the first proposed solution showing this idea, which taints all the input to applications, and enforces a runtime checking to see whether the untrusted data is used in a security-sensitive way. Inspired by Perl’s taint analysis, many approaches then have been proposed (e.g., [25,26,27,28]). One of them for particularly detecting format-string attacks is TypeQualifier presented by Shankar et al. [17]. In their approach, if the potentially tainted data is used as the format string, then an error is issued. From this point, it seems very similar to ours. However, these two approaches are based on different mechanism to implement (static analysis for this technique, run-time tracking in ours). Besides, this technique requires programmers’ efforts to specify which object is tainted, and consequently presents an additional burden on developers. In contrast, our approach appears almost transparent. In addition, this approach is more conservative than necessary because static analysis is inherently limited and much supposedly tainted data is actually perfectly safe, whereas our approach is not so conservative since we only prevent those tainted data which contains dangerous format specifier and is used as a format string.

Run-Time Techniques. In contrast to compile-time analysis, run-time techniques present a low burden on developers and uniformly improve the security assurance of applications. Libformat [13] (a preloaded shared library) is a case of such examples, which aborts any processes if they call `printf`-family functions with a format string that is writable and contains `%n`-directive. This technique is effective in defending against real format-string attacks, but in most cases both writable format strings and `%n`-directives associated destination are legal, and consequently it generates many false alarms. Despite the similarity of both our approach and Libformat guard the writable and `%n`-directive contained in format-string argument, ours is not so much conservative as this technique, and we provide more flexibility.

Another notable run-time approach is referred as Libsafe [14], which implements a safe subset of format functions that will abort the running process if the inspected

corresponding pointer of `%n` directive points to a return address or frame pointer. This approach also looks very close to ours. However, we should note the major difference is Libsafe provides every format string (despite static or not) checking on limited scope (i.e., return address and frame pointer), whereas we only check apparently-tainted areas to identify the root-cause (untrusted data) of format string attacks. Thus, as shown in our experiment, Libsafe would sometimes lead to false negatives, while our approach could catch almost all of them.

The idea of using dynamic taint analysis for detecting security attacks was attracted a lot of attention. Suh et al. [29] proposed a *dynamic information flow tracking* scheme to protect programs against malicious attacks. Chen et al. [30] developed a *pointer taint-ness* detection architecture to defeat the most memory corruption attacks. These two approaches were demonstrated useful and efficient, but both of them require processor modifications to support taint-tracking. Unlike these two hardware solutions, Newsome et al. proposed a software approach, TaintCheck [31], to monitor and control the program execution at a fine-grained level. While this approach is very promising and can defend against a large number of security attacks with fewer false positives, the main drawback is that it incurs significant performance overhead by a factor of 10 or more because of its emulator-based implementation. Our approach follows their way in dynamic taint but differs in the granularity and interception.

Combined Compile-Time/Run-Time Approaches. FormatGuard [16] is an extension to `glibc` that provides argument number checking for `printf`-like functions with the support of GNU C Preprocessor. Programs need to be recompiled but without any modifications for its protection. Although FormatGuard can protect the `printf`-like functions efficiently, it cannot protect the format functions which use `vararg` such as `vprintf` (in this case it is not possible to count the actual number of parameters at compile time). Besides, FormatGuard may result in false negatives when another format specifier is replaced with `%n`-directive.

White-listing [15] is another approach which tries to achieve the benefits of both static and run-time techniques. By cleverly using a source code transformation, this approach automatically inserts the code that maintains and checks against the white-listing containing safe `%n`-writable address ranges via the knowledge gleaned from static analysis. White-listing gains high precision with very few false negatives and few false positives, and imposes little performance overhead. However, one limitation of this approach is that applications which are only re-compiled using White-listing can benefit from its protection.

7 Conclusion and Future Work

In this paper, we have proposed a practical and transparent approach to the detection and prevention of format-string attacks. We exploit the dynamic taint analysis and library interpositions technique, which allow us to protect the software without any recompilation, to achieve our goals. Through the thorough analysis and empirical evaluation, we show that our approach has very few false negatives and false positives, and just imposes a little performance overhead on the protected software.

Due to the similarity to format-string attacks, SQL injection is another dangerous attack caused by unvalidated input. We feel our approach is also applicable for the prevention of this attack, for instance, we can taint the input data and check against SQL syntax to see if these data represent an invalid user input. One of our future work will apply our approach to deal with this attack. Other future work includes to port our approach to other platforms (e.g., Windows), to investigate attack repair approaches and so on.

Acknowledgements

We thank Michael F. Ringenburg, Erwang Guo, Yi Wang, Yi Ge and anonymous reviewers for their support during this work. This research was supported in part by Chinese National Science Foundation under grant 60373064 and Chinese National 863 High-Tech Program under grant 2003AA144010.

References

1. "tf8". Wu-Ftpd Remote Format String Stack Overwrite Vulnerability. At <http://www.securityfocus.com/bid/1387>. (2000)
2. NIST National Vulnerability Database. At <http://nvd.nist.gov>. (2006)
3. Scut, team teso:Exploiting Format String Vulnerabilities. At <http://www.team-teso.net/releases/formatstring-1.2.tar.gz>. (2001)
4. Riq and Gera. Advances in format string exploitation. *Phrack Magazine*, 59(7). At <http://www.phrack.org/phrack/59/p59-0x07>. (2002)
5. Lhee, K. and Chapin, S.:Buffer overflow and format string overflow vulnerabilities. *Software-Practice & Experience*. Vol 33(5). Pages: 423-460. (2003)
6. Anley, C.:Advanced SQL Injection In SQL Server Applications. Technical Report, NGSSoftware Insight Security Research. (2002)
7. Jacobowitz, D.:Multiple Linux Vendor rpc.statd Remote Format String Vulnerability. At <http://www.securityfocus.com/bid/1480>. (2000)
8. Kaempf, M.:Splitvt Format String Vulnerability. At <http://www.securityfocus.com/bid/2210/>. (2001)
9. NSI Rwhoisd Remote Format String Vulnerability. At <http://www.securityfocus.com/bid/3474>. (2001)
10. Pelat, G.:PFinger Format String Vulnerability. At <http://www.securityfocus.com/bid/3725>. (2001)
11. Goldsmith, D.:TCPflow Format String Vulnerability. At <http://www.securityfocus.com/bid/8366>. (2003)
12. Xiao, Z.:An Automated Approach to Software Reliability and Security. Invited Talk, Department of Computer Science, University of California at Berkeley. (2003)
13. Robbins, T.:Libformat. At <http://www.wiretapped.net/~fyre/software/libformat.html>. (2001)
14. Tsai, T. and Singh, N.:Libsafe 2.0: Detection of Format String Vulnerability Exploits. At <http://www.research.avayalabs.com/project/libsafe/doc/whitepaper-20.pdf>. (2001)
15. Ringenburg, M. and Grossman, D.:Preventing Format-String Attacks via Automatic and Efficient Dynamic Checking. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, Alexandria, Virginia. (2005)

16. Cowan, C., Barringer, M., Beattie, S. and Kroah-Hartman, G.:FormatGuard: Automatic protection from printf format string vulnerabilities. In *Proceedings of the 10th USENIX Security Symposium (Security'01)*, Washington DC. (2001)
17. Shankar, U., Talwar, K., Foster, J. S., and Wagner, D.:Detecting format string vulnerabilities with type qualifiers. In *Proceedings of the 10th USENIX Security Symposium (Security'01)*, Washington DC. (2001)
18. TIS. Executable and Linkable Format Version 1.1. At <ftp://download.intel.com/perftool/tis/elf11g.zip>
19. Cormen, T., Stein, C., Rivest, R. and Leiserson, C.:*Introduction to Algorithms*. MIT Press, second edition. (2002)
20. Smirnov, A. and Chiueh, T.:DIRA: Automatic Detection, Identification and Repair of Control-Hijacking Attacks. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, San Jose, CA. (2005)
21. Avijit, K., Gupta, P., and Gupta, D.:TIED, LibsafePlus: Tools for Runtime Buffer Overflow Protection. In *Proceedings of the 13th USENIX Security Symposium (Security'04)*. (2004)
22. DeKok, A.:PScan: A limited problem scanner for C source files. At <http://www.striker.ottawa.on.ca/~aland/pscan/>. (2000)
23. The GNU Compiler Collection. Free Software Foundation. At <http://gnu.gcc.org/>
24. Perl security manual page. At <http://www.perldoc.com>.
25. Zhang, X., Edwards, A. and Jaeger, T.:Using CQual for static analysis of authorization hook placement. In *Proceedings of the 11th USENIX Security Symposium (Security'02)*. (2002)
26. Foster, J., Fahndrich, M. and Aiken, A.:A theory of type qualifiers. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'99)*. (1999)
27. Evans, D. and Laroche, D.:Improving Security Using Extensible Lightweight Static Analysis. In *IEEE Software*, January/February. (2002)
28. Tuong, A.N., Guarnieri, S., Greene, D., Shirley, J. and Evans, D.:Automatically hardening web applications using precise tainting. In *Proceedings of the 20th IFIP International Information Security Conference (SEC'05)*. (2005)
29. Suh, G., Lee, J., Zhang, D. and Devadas, S.:Secure program execution via dynamic information flow tracking. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'04)*, Boston, MA. (2004)
30. Chen, S., Xu, J., Nakka, N., Kalbarczyk, Z. and Iyer, R. K.:Defeating memory corruption attacks via pointer taintedness detection. In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN'05)*. (2005)
31. Newsome, J. and Song, D. :Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, San Jose, CA. (2005)

Low Latency Anonymity with Mix Rings

Matthew Burnside and Angelos D. Keromytis

Department of Computer Science, Columbia University
{mb, angelos}@cs.columbia.edu

Abstract. We introduce *mix rings*, a novel peer-to-peer mixnet architecture for anonymity that yields low-latency networking compared to existing mixnet architectures. A mix ring is a cycle of continuous-time mixes that uses carefully coordinated cover traffic and a simple fan-out mechanism to protect the initiator from timing analysis attacks. Key features of the mix ring architecture include decoupling path creation from data transfer, and a mechanism to vary the cover traffic rate over time to prevent bandwidth overuse. We analyze the architecture with respect to other peer-to-peer anonymity systems – onion routing and batching mixnets – and we use simulation to demonstrate performance advantages of nearly 40% over batching mixnets while protecting against a wider variety of adversaries than onion routing.

1 Introduction

In 1981, Chaum proposed a method for anonymous communication called a mixnet[3]. Chaum envisioned a network of proxies, called mixes, each with a well-known public key. In this system, when an *initiator* wishes to send a message anonymously to a *responder*, the initiator chooses a route through the mixes and wraps the message in a series of encrypted layers, one layer per mix in the route. Each layer contains next-hop information and the encrypted next-innermost layer; the layers are peeled off one by one as the message traverses the route to the responder. Chaum’s original system has descendants in a wide variety of anonymity-providing systems, including systems for anonymous web surfing[18, 2], peer-to-peer anonymity[11], and network level anonymity in the form of onion routing[12, 10].

Onion routing, equivalent to Chaum’s system as described thus far, yields low-latency connectivity but is vulnerable to timing analysis. An adversary observing all links in an onion routing network can record arrival and departure times for all messages in the network and use statistical methods to determine exactly who is communicating with whom[19, 17]. Mixnets go further and attempt to address this vulnerability by processing messages in batches to disguise timing correlations. However, each mix along a route must delay processing a batch until the batch is full, so, in practice, mixnet communication is generally slower than onion routing. The result is, broadly, two designs: one provides strong anonymity at the expense of performance, the other provides weaker anonymity with performance.

We introduce an intermediate solution, *mix rings*, that give better performance than mixnets while maintaining anonymity that is stronger than onion routing. The goal of a mix or onion router is to disguise the correspondence between the input and output messages that each relays. This goal is achieved through cryptographically transforming each message so an observer cannot link the messages and then, in the case of mixes, obscuring timing information either through batching or individually delaying the messages. The work presented here introduces a third mechanism for obscuring timing information that eliminates the need for batching (and the corresponding performance degradation that comes with batching) while maintaining the anonymity properties of the standard mixnet.

The intuition is that the batching process in a continuous-time mixnet inherently reduces performance, since a message has to be delayed at each mix until the batch is full before it is forwarded. Batching is used to protect against timing analysis, and we show that the mix ring architecture protects against the same attacks, while paying a reduced performance penalty.

We route cover messages in a ring of mixes that contains the initiator. Using mechanisms described later, the source (*i.e.*, the initiator) of these cover messages is hidden, so an outside observer only sees an unbroken stream of messages traversing the ring, with no obvious source or destination. When the initiator wants to send a message to the responder, the initiator replaces a cover message with a data message that “forks” (we will use the term *fan-out*) into two messages when it reaches an arbitrary point on the ring. One of the messages contains the data destined for the responder, the other is a cover message that continues around the ring. Since all traffic movement in the ring is coordinated, timing analysis can determine no more than that the message originated somewhere in the ring. We demonstrate that this system gives performance advantages of nearly 40% over batching mixnets, while providing a stronger form of anonymity than onion routing.

Note that there exists a form of mixnet, called a mix cascade, that similarly provides low latency traffic. A mix cascade consists of a dedicated set of servers that redirect traffic from a large set of users on a predefined route. There is considerable discussion in the field of anonymity research on the merits of mix cascades vs. peer-to-peer style anonymity systems; [9] contains an excellent summary of both sides of the discussion. Mix rings are a peer-to-peer network, and thus we compare them only with other peer-to-peer anonymity systems.

The rest of this paper is organized as follows: in Section 2, we discuss related work. In Section 3, we define our threat model and give a detailed explanation of our system. In Sections 4 and 5, we examine the anonymity and performance of mix rings. We conclude the paper in Section 6.

2 Related Work

Systems based on onion routing include Tarzan[11], a peer-to-peer, IP-layer anonymizing network and Tor[10]. Tor extends onion routing by adding support

for integrity protection, congestion control, and location-hidden services through rendezvous points. Batching mixes add protection from timing analysis attacks by pooling messages in batches. Mixes have been used in a wide variety of applications such as ISDN service[15], IP-layer infrastructure[14], and email[6].

Stop-and-go mixes[13], or *sg-mixes*, come from a subtype of mixes called *continuous-time* mixes. Messages in a continuous-time mixnet are delayed individually, rather than in batches. In *sg-mix* networks, message delays are randomly chosen from an exponential distribution within a window of estimated arrival time for that mix. Of course, this requires the initiator to predict traffic levels throughout the mixnet. Mix rings are closely related to *sg-mixes* but use constant rather than random delays so a principal does not have to predict network traffic levels. (There exist attacks on connection based continuous-time mixes that are based on observation of injected traffic[5], but these attacks do not apply to mix rings, as mix rings are not connection-based.)

Pipenet[4] is effectively a mix network with the addition of pairwise cover traffic between all mixes. This network provides strong theoretical anonymity but is impractical due to its considerable bandwidth requirements. Mix rings are similar to Pipenet but we reduce bandwidth requirements through mix ring's ability to increase and decrease cover traffic on demand.

Fitzmann, *et al.*, consider anonymity in a ring topology in the context of local area networks [16] and describe such a system as being both fault tolerant and efficient. Mix rings may be considered an extension of this work into the inter-network realm. Other ring-related work includes [1] which uses tokens to carry messages on routes that they compare to public transportation bus routes.

Danezis, *et al.* route dummy messages in a ring through an anonymity system in what they call heartbeats [8]. Using heartbeats, an anonymity system can detect some forms of tampering on the network, including an adversaries who might speed up or slow down network links, or block them entirely. Mix rings inherently provide the same protections, since every cover traffic message in the ring can be viewed as a heartbeat.

3 Design and Operation

We begin our discussion by describing our threat model and the design goals for the system, then move on to the design of the system.

3.1 Threat Model

We consider the broad categories of adversaries first described in [18]: a local eavesdropper (*e.g.*, a malicious first-hop router), one or more malicious mixes, a malicious responder, and finally a global adversary who can observe and modify all network links. We further assume that the global adversary can compromise some, but not all, mixes in the network. We assume no adversary can reverse cryptographic transformations.

3.2 Design Goals

Our broad goal is to design a peer-to-peer system that finds a compromise between the (relatively) weak anonymity/low latency of onion routing and the strong anonymity/high latency of mixnets. Thus, the specific goals of the architecture proposed in this paper are to provide the flexibility of a peer-to-peer anonymity system, while demonstrating anonymity comparable to that of a mixnet, but with lower latency than a mixnet. We specifically consider latency rather than bandwidth due to the batching nature of mixnets; the batching process affects latency but does not inherently affect bandwidth.

3.3 System Model

The basic mix ring infrastructure is a set of continuous-time mixes, each with a well-known public key. (Note that in continuous-time mixes, messages are delayed individually, rather than in batches.) The initiator must be a mix, but the responder need not be.

When a mix receives a message, it decrypts the message to obtain next-hop information, a set of options, and a recursively defined body. (See Section 3.4 for more details on the structure of a message.) The mix performs any operations specified by the options, pads the body out to the original length of the message, and then forwards it on to the specified next hop.

Consider an initiator who wants anonymous communication with a particular responder. This initiator chooses a random set of mixes, logically organizes them into a ring, and negotiates session keys with each. The initiator then uses the method described in Section 3.5 (below) to anonymously start routing cover traffic around the ring.

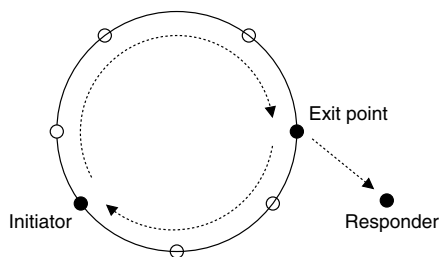


Fig. 1. The initiator routes both cover traffic and data traffic around the ring. The data traffic leaves the ring through a randomly chosen exit point.

The initiator sends a message to the responder, outside the ring, by constructing a special message that fans out when it reaches a randomly chosen exit point on the ring. That is, when the message is processed by the exit point, the processing generates two messages: one message destined for the responder and one message that continues along the ring, indistinguishable from other cover

traffic. The result is that the total number of cover messages circling the ring is unchanged, and from an adversary’s point of view, each member of the ring is equally likely to have been the source of the message. (See Figure 1.)

3.4 Mix Ring Messages

Messages in mix rings are similar to other forms of onion-style messaging, *e.g.*, Minx[7]. Each message has the following form:

$$\{T, H_1, B_1, [H_2, B_2], P\}$$

A message consists of a set of options T (defined below), a header H_1 containing the next-hop IP address and port, and a recursively defined body B_1 encrypted using the next hop’s session key. The message may contain an optional second header and body, H_2 and B_2 , in the case of a fan-out message. Finally, as each layer of the message is removed, padding P is appended so the total message length remains constant.

We define the following options T on a message:

- set-rate.** A *set-rate* option has a field that indicates the rate at which the mix should forward every subsequent message.
- delay.** A *delay* option instructs a mix to wait for some time period d before processing the rest of the message, including any other options that may be set.
- fan-out.** A message with the *fan-out* option set contains in its body two next-hop messages, H_1, B_1 and H_2, B_2 . Typically, one of these next-hop messages is standard cover traffic which is forwarded to the next hop in the ring; the other is routed to the responder.

Each option may be set in any of the encrypted layers in a message, and a single layer may have multiple options. Note that *set-rate* applies to all future messages received by the mix, whereas *delay* is specific to the message layer containing that option.

3.5 Constructing the Ring

An initiator μ_0 uses a trusted, well-known, directory server (similar to the Tor[10] directory server) to arbitrarily select a set of mixes $\mu_1, \mu_2, \dots, \mu_n$ that will comprise the ring. The initiator then selects a second set of mixes $\nu_1, \nu_2, \dots, \nu_n$ which form a tunnel through the mixes to μ_i , an arbitrary mix in μ . Through this tunnel, the initiator negotiates a session key with each mix in the ring be used for encrypting the layers of each message (see Figure 2).

Note that when a mix is sent a message with no option set (as are the mixes in the tunnel) it behaves as a standard continuous-time mix. The purpose of the tunnel at startup is to protect against a local adversary (an adversary situated on the initiator’s local network) who can otherwise observe the initiator constructing the ring. See Section 4.1 for further discussion.

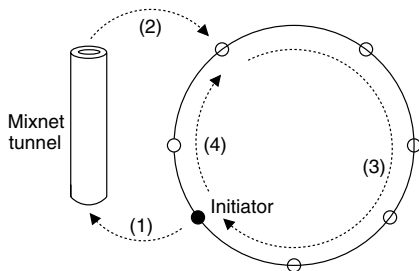


Fig. 2. Injecting cover traffic into the ring. The initiator tunnels all cover traffic (1) to an arbitrarily chosen mix on the ring (2), from which the traffic is routed around the ring (3). When the traffic returns to the initiator, it is replaced with cover traffic routed directly through the the ring (4), rather than through the tunnel.

Now with a session key for each mix in the ring, the initiator begins transmitting cover traffic at some rate r , routed through the tunnel and around the ring, destined for the initiator itself. The first of these messages has the *set-rate* option defined on each of its layers, instructing each mix on the default rate r to transmit each subsequent message. As each cover message returns to the initiator, it is seamlessly replaced with a new cover message routed through μ_1 rather than the tunnel. When the desired number of cover messages has entered the ring, the initiator closes the tunnel. The desired number of cover messages is a factor of the bandwidth requirements of the initiator and the bandwidth limitations of the links on the ring itself. It can be as small as one, or as large as the network infrastructure of the ring can handle.

Reiter *et al.* showed that all anonymity systems must reset periodically to allow new nodes to join[18, 21]. We use the standard network time protocol (NTP) to coordinate periodic resets across the mix ring system for this purpose. We assume that, after each reset, every participating mix builds a new ring. This protects from an adversary observing the initiator’s local network who might otherwise be able to identify the initiator by observing which mix is the first to send traffic (since this would have to be the initiator). With the above method, that same adversary can only determine that the initiator is a participant in one or more rings, but not that he or she is a given ring’s creator.

At this point, the construction of the ring is complete. The cover traffic is flowing around the ring, each message advancing at a rate of r . This is the default behavior of the ring when no message is being sent.

3.6 Default Behavior

The default behavior occurs when no message is being sent. At some rate r the initiator sends cover messages through the ring. As each cover message traverses the ring and returns to the initiator, it is seamlessly replaced with a new one. If

a cover message fails to arrive within some window of its appointed time, or if the messages arrive out of order, the initiator considers such messages lost and does *not* transmit a new cover message; instead it uses the method described in Section 3.7 to refill these gaps in the cover traffic.

If the initiator filled each gap as it appeared, an adversary could drop one or more cover messages and then observe at what point the messages were replaced; whichever mix replaced the messages would be the initiator. Thus, we use another method, described below.

3.7 Filling Gaps in the Cover Traffic

When the initiator discovers a gap in the cover traffic, it cannot simply insert a new cover message because it would be clear to an adversary that the mix which filled the gap was the initiator. Instead, the initiator chooses a random mix μ_j on the ring and instructs *that* mix to fill the gap. This instruction is performed using tools already described: on the gap's second trip around the ring, the initiator replaces the message immediately preceding it with a message constructed in the following manner. In the message layer corresponding with μ_j , the initiator sets the *fan-out* flag. The message headers H_1 and H_2 have the same next-hop destination (the next mix in the ring) but each is set with a different *delay* value, one $d = r$ and one $d = 2r$, such that the gap is filled when the second message is relayed. For larger gaps, this procedure is repeated as necessary.

3.8 Varying the Traffic Rate

Under normal circumstances, the traffic rate in the ring is set to a low volume to avoid consuming unnecessary bandwidth. Only when the initiator wishes to send traffic is the rate increased. When the initiator wishes to increase or decrease the traffic rate, it constructs a message to each mix in the ring with carefully calculated *delay* and *set-rate* values such that all mixes in the ring execute the *set-rate* at approximately the same time (this instruction actually takes the form of a single message, with each layer in the message instructing that particular hop on its rate and delay values). Perfect coordination is not required so long as the initiator's rate change is not statistically unique with respect to the rate changes of the other mixes.

Over time, due either to properties of the network or actions of an adversary, the cover traffic may become unevenly distributed around the ring. That is, the traffic may become grouped together in "clumps." We do not consider it further here, but in future versions of the system, the initiator may counter this action through judicious use of the *delay* option.

3.9 Sending a Message

To send a message, the initiator chooses a random member of the ring μ_e as the exit point for that message. It then constructs a message which is identical to

the other cover messages except that the layer of the message destined for μ_e has the *fan-out* option set. One branch of the fan out, the data, is destined for the responder; the other, cover traffic, is routed to the next hop on the ring. The initiator can vary the ratio of data to cover messages to increase or decrease the traffic bandwidth.

3.10 Receiving a Message

The current mix rings implementation does not include a mechanism for a return channel. However, we include in this section a draft for this mechanism which will be built in future work. Briefly, the initiator includes in the message a reply block which the responder may use to send messages back to the initiator. The reply block is essentially an empty onion, designed to travel along the return route of the ring. The responder fills in the body of the message and sends it to the first hop of the return path (*i.e.*, the exit point on the ring). When the initiator receives the message, the return path is complete, but the initiator must generate a new cover message that continues along the ring in place of the return message. Otherwise an adversary could simply observe the path of return message to identify the initiator.

4 Analysis

In this section, we analyze the mix ring architecture by comparing it with onion routing and mixnet architectures. We show performance advantages of nearly 40% over batching mixnets and demonstrate a stronger form of anonymity than strict onion routing.

Table 1. Comparison of initiator exposure in onion routing, mix rings, and mixnets. The columns represent, respectively, a malicious first-hop router, a malicious intermediate mix or set of mixes, and a malicious responder.

	Local eavesdropper	Bad mixes	End host
Onion routing	Exposed	No	No
Mix rings	Possibly	No	No
Mixnets	Possibly	No	No

Table 1 gives a comparison of the three architectures with respect to various adversaries, each of which will be analyzed further in the remainder of this section. We examine attack vectors by breaking them into three classes: adversaries at or near the initiator (a local eavesdropper), adversaries in the middle of the network (compromised mixes), and adversaries at or near the responder. We then consider some specific attacks against the cover traffic in the ring, and attacks by a global adversary.

4.1 Attacks from a Local Eavesdropper

In onion routing, an initiator is immediately exposed to an adversary of this type, since the adversary can observe the construction of the onion route. In mixnets, the adversary can detect initiator activity, but generally cannot distinguish whether the initiator is truly an initiator or simply an intermediate node on a mix route. Initiator activity in mix rings is identical to that of mixnets – the initiator is simply building up mix routes, and thus the initiator is no more or less exposed than an initiator in a mixnet.

4.2 Attacks from Compromised Mixes

In both onion routing and mixnets, a malicious intermediate mix can identify its predecessor and successor on a given route. Also, if the responder is a node outside the network and the compromised mix is the last hop on the route before the responder, the adversary is trivially able identify the responder.

Similarly, a malicious mix in a ring can identify its predecessor and successor. It has knowledge of the traffic rate in the ring, and whether or not it is an exit point. If the mix is an exit point, it can identify the responder.

When a compromised mix knows it is on a route of interest, it can record its predecessor on that route. Since the initiator has to be on the route between itself and the responder after every reset, while all other mixes will only be on that route with some probability less than one, over time the initiator will be the predecessor of the compromised mix more often than any other mix. This attack is called the predecessor attack and was first proposed by Reiter and Rubin[18].

Of course, the compromised mix must identify when it is in a route of interest; this is only achieved when multiple compromised mixes are on the same route. In onion routing, if a compromised mix is the exit point for a route, it uses timing analysis to identify whether any of its collaborators are also on that route. If there is one, then the collaborator records its predecessor. In this case, Wright, *et al.*, showed that after $O(n^2 \ln n)$ resets, the adversary can with high probability identify the initiator[20].

The mixnet attack is similar, but requires all mixes in a route of length l be compromised before recording a predecessor data point. Wright showed that mixnets survive for $O(n^l \ln n)$ resets before the adversary can identify the initiator with high probability .

Under the predecessor attack, mix rings behave as mixnets. The portion of the ring past the fan-out is irrelevant since the attack takes place over the portion of the ring between the initiator and the exit point. This is a segment of the ring and a segment of the ring is also a mix route. If the distance from initiator to exit point is i , then the initiator in a mix ring is compromised after $O(n^i \ln n)$ resets.

4.3 Attacks from the Responder

Consider a ring of circumference $2n$ and an onion or mix route of length n . (These are comparable since, on average, assuming randomly chosen exit points,

a message will traverse n mixes before it reaches the exit point and leaves the ring.) A malicious responder in a standard onion or mix route must unroll the route, compromising each of the n mixes in turn, to identify the initiator. Similarly, a malicious responder in a mix ring must, on average, compromise n mixes from the exit point back to the initiator to make positive identification. Mix rings, mixnets and onion routing are equally strong against attacks of this form from the responder.

4.4 Attacking the Cover Traffic

Cover traffic is a key difference between mix rings and mixnets. The cover traffic in a mix ring is cryptographically transformed in the same fashion as the data-carrying traffic. An adversary is therefore unlikely to be able to distinguish cover traffic from data traffic without reversing the cryptographic transforms. The adversary can, however, attempt to inject new cover traffic, or drop, delay, modify or replay.

New cover traffic introduced by the adversary will be detected and dropped at its first hop since we assume only the initiator possesses the session key required to construct a valid message. However, if the adversary cuts an arbitrary link for a short time but does not drain the cover traffic completely, then the mechanism described in Section 3.7 is sufficient to fill any gaps. Meanwhile, the remaining cover traffic is still useful for carrying data. If the adversary drains the cover traffic completely, the ring must be reconstructed as in Section 3.5. This is effectively a denial of service.

If, instead, the adversary attempts to delay the traffic (this may also occur due to properties of the underlying network) the cover traffic may become unevenly distributed around the ring. This is handled as described in Section 3.8.

4.5 Attacks from a Global Adversary

Similar to mix nets and onion routing, a global adversary is able to observe the construction of the ring, and this compromises the initiator from the start.

4.6 Further Analysis

An important consideration in mix rings is that there must be members of the community willing to operate nodes in a mix ring in which they, themselves, are not the initiator. Given that there are members of the community who have shown willingness to operate Tor and Mixminion nodes, for example, we believe it a reasonable assumption that like-minded individuals would also be willing to operate nodes in a mix ring.

5 Performance

To compare the performance of onion routing, mix rings, and mixnets, we designed and deployed a simulator on PlanetLab, a worldwide experimental

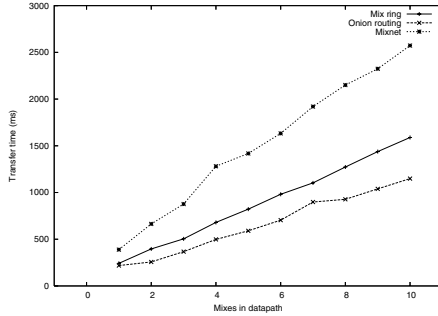


Fig. 3. Transfer time for a 50kb file over varying route lengths

overlay network. The simulator is designed to emulate onion routing, mix rings, and mixnets. It is important to note that we only consider peer-to-peer anonymity systems; we do not consider mix cascades. The simulator is a daemon that runs on each node. At startup, the daemon is configured to operate as a node in an onion routing system, a mix ring system, or a mixnet system. The difference between these modes of operation consist primarily of batching for mixnets and handling the message options for mix rings. More specifically, the daemon behaves as follows.

- When configured as an onion-router, the daemon listens for messages on a pre-defined port. On receipt of a message m , the daemon decrypts the message using its private key to obtain next-hop information. The message is then immediately forwarded to the next hop.
- When configured as a router in a mix net, the daemon collects messages arriving on a pre-defined port until the specified number of messages has arrived (the batch size). Then each message is decrypted to obtain next-hop information and forwarded onward.
- Finally, when configured as a node in a mix ring, the daemon behaves as an onion router, but obeys any options that may be set in the message (see Section 3.4).

We deployed the simulator on approximately 30 PlanetLab nodes. For testing mixnets and onion routing, we chose six nodes at random to behave as onion routers or mixes, respectively. In the case of mix rings, we chose twelve of the nodes at random, since in a mix ring a one-way message only traverses half the ring, on average. We configured the mixnet with a batch size of 10 messages; we configured the mix ring with a default cover traffic rate of 10ms between messages. Each message is 8192 bytes.

In each of the following tests, we chose one node as the initiator and a node outside the route or ring as the responder. The measurements do not take into account interactions with other users, nor do they compare startup times; these will be the subject of future analysis.

Figure 3 compares the transfer time of a 50kb file under each of the three schemes as the length of the route increases. In the case of the mix ring, the exit point on the mix ring was pegged at the $\frac{n}{2}$ node. It is clear that, while the mix ring does not outperform onion routing, it has considerably lower latency than the mixnet, on the order of 40% improvement for a given path length.

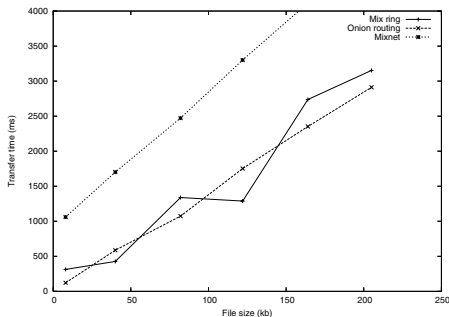


Fig. 4. Transfer time as a function of file size in a 6-node mix route, 6-node onion route, and 12-node ring

Figure 4 compares the three schemes as the file size increases. In this case, the mix ring exit point is chosen randomly, so the variance of the mix ring plot has increased due to the randomly chosen exit points. However, the trend is still clear: mix ring performance is considerably better than mixnets and approaches that of onion routing.

6 Conclusion

This paper presents mix rings, an anonymity system stronger than onion routing, with better performance than mixnets. Mix rings are a compromise between the weaker anonymity/low latency of onion routing and the strong anonymity/high latency of mixnets. Mix rings use cover traffic to protect against timing analysis and provide a mechanism to vary the cover traffic rate to prevent bandwidth overuse. We show that mix rings provide anonymity that is stronger than onion rings, and comparable to mixnets. For file transfers, we show that mix rings exhibit nearly 40% performance improvements over mixnets. This work makes strong anonymity a possibility for low-latency applications that may not have previously had the option available.

In future work, we will consider bidirectional traffic, perhaps using reply blocks. We will also give further study to the bandwidth interactions between multiple rings.

Acknowledgments

This research was supported by NSF grant NSF CCF-05-41093.

References

- [1] A. Beimel and S. Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1):25–39, 2003.
- [2] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.
- [3] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
- [4] W. Dai. Pipenet 1.1. Usenet post, August 1996.
- [5] G. Danezis. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2004)*, LNCS, May 2004.
- [6] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
- [7] G. Danezis and B. Laurie. Minx: A simple and efficient anonymous packet format. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004.
- [8] G. Danezis and L. Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.
- [9] C. Díaz, G. Danezis, C. Grothoff, A. Pfitzmann, and P. F. Syverson. Panel discussion - mix cascades versus peer-to-peer: Is one concept superior? In *Privacy Enhancing Technologies*, pages 242–242, 2004.
- [10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
- [11] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.
- [12] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.
- [13] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-Go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.
- [14] The NymIP Effort. <http://nymip.velvet.com>.
- [15] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.
- [16] A. Pfitzmann and M. Waidner. Networks without user observability – design options. In *Proceedings of EUROCRYPT 1985*. Springer-Verlag, LNCS 219, 1985.

- [17] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.
- [18] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.
- [19] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *Proceedings of ESORICS 2003*, October 2003.
- [20] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium (NDSS '02)*. IEEE, February 2002.
- [21] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

Breaking Four Mix-Related Schemes Based on Universal Re-encryption

George Danezis

K.U. Leuven, ESAT/COSIC,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium
`George.Danezis@esat.kuleuven.be`

Abstract. Universal Re-encryption allows El-Gamal ciphertexts to be re-encrypted without knowledge of their corresponding public keys. This has made it an enticing building block for anonymous communications protocols. In this work we analyze four schemes related to mix networks that make use of Universal Re-encryption and find serious weaknesses in all of them. The Universal Re-encryption of signatures is open to existential forgery, and the two mix schemes can be fully compromised by an passive adversary observing a single message close to the sender. The fourth scheme, the rWonGoo anonymous channel, turns out to be less secure than the original Crowds scheme, on which it is based. Our attacks make extensive use of unintended ‘services’ provided by the network nodes acting as decryption and re-routing oracles. Finally, our attacks against rWonGoo demonstrate that anonymous channels are not automatically composable: using two of them in a careless manner makes the system more vulnerable to attack.

Keywords: Universal re-encryption, re-encryption mix networks, anonymous communications, traffic analysis.

1 Introduction

An important technique to achieve anonymous communication is the *mix*, an anonymizing relay, first proposed by David Chaum [1]. In his scheme messages to be anonymized, on their journey from Alice to Bob, are first encrypted under the public keys of all intermediate mixes. The messages are then relayed by all mixes in succession that decrypt them, effectively peeling off a layer of encryption at the time, and forwarding them to the next mix. As a result an observer of the network should find it hard to link senders and receivers of messages.

Many mix based systems, inspired from this architecture, have been designed and deployed [2,3,4]. They all use a hybrid encryption scheme, that combines the necessary public key cipher with a symmetric key cipher for bulk encryption. This technique keeps the computational cost of running a mix low, and allows more messages to be mixed together. Yet this architecture suffers from *replay* attacks: the same message, if routed twice in the mix network, will at each stage decrypt to bitwise exactly the same plaintext. To prevent adversaries from

making use of this property to facilitate traffic analysis, most schemes keep track of the messages processed and refuse to process them again. The storage cost is proportional to the number of messages processed.

An alternative approach – also with the independent advantages of proofs of robustness – relies on mixed messages being re-encrypted instead of decrypted. In such schemes [5,6] messages are encrypted using the El-Gamal public key cipher [7], and each mix node re-encrypts them on their way. Finally all messages are decrypted by a threshold decryption scheme at the end of the route. The re-encryption is randomized, and replaying a message will lead to different intermediate messages in the network. The re-encryption operation does not require any secrets, but requires the knowledge of the public key used for encryption.

Golle *et al.* [8] proposed a scheme, named Universal Re-encryption, that does away with the requirement to know the public key, under which a ciphertext was encrypted, to be able to re-encrypt it. A plaintext m encrypted under public key (g, g^x) has four components (using fresh k, k'):

$$\text{URE}_x(m) := (a, b, c, d) := (g^{k'}, (g^x)^{k'}; g^k, (g^x)^k \cdot m) \quad (1)$$

Such a ciphertext can be re-encrypted by anyone, and become unlikable to the original one using fresh z, z' :

$$(a', b', c', d') := (a^{z'}, b^{z'}; a^z \cdot c, b^z \cdot d) \quad (2)$$

Note that the re-encrypted product of Universal Re-encryption is a valid ciphertext of message m , encrypted under the secret key x , i.e. $\text{URE}_x(m)$.

The Universal Re-encryption primitive itself, and its extensions [9], are believed to be secure. In this work we study the applications of this primitive, in the context of anonymous communications, that turn out to have numerous weaknesses.

First we demonstrate that the attempt of Klonowski *et al.* [10] to make re-encryptable RSA signatures is insecure, and vulnerable to existential forgery. Then we consider the mix scheme of Klonowski *et al.* [11] and Gomulkiewicz *et al.* [12] that attempt to use Universal Re-encryption to build replay resistant mix networks. Their schemes can be attacked by a passive adversary that observes the message ciphertext at just one point, close to the sender Alice. Finally we consider the rWonGoo scheme by Lu *et al.* [13]. The scheme takes into account that the careless use of Universal Re-encryption is susceptible to tagging attacks, and a variant of re-encryption is used. Yet rWonGoo fails to protect against all attacks, and we demonstrate that it is in fact weaker than the simple Crowds [14] anonymity scheme. We propose a fix to make rWonGoo as secure as Crowds, yet the heavy cryptography used becomes superfluous.

2 Breaking the “Universal Re-encryption of Signatures”

Klonowski *et al.* [10] extend the universal re-encryption scheme by Golle *et al.* [8], that allows ElGamal [7] ciphertexts to be re-encrypted along with a valid

RSA [15] signature. The transform is key less, and can be performed by any third party. The key feature of the Klonowski *et al.* scheme is that the signature associated with the ciphertext remains valid, despite the ciphertexts being modified through re-encryption. Schemes with such properties have the potential to be used in anonymous credential, e-cash and electronic election schemes, as well as a plethora of other application in the field of privacy enhancing technologies. Unfortunately their scheme is insecure since signed ciphertexts can be combined, without the knowledge of any signing secrets, to produce valid signatures.

Assuming that $N = pq$ with p and q being two random large primes and let g be in \mathbb{Z}_N^* . All operations are performed modulo N , unless otherwise stated. For a message m an authority creates an RSA signature m^d (d being its signature key). To encrypt the message to a public key $y = g^x$, the authority chooses uniformly at random two values k_1 and k_2 . A cipher text in the Klonowski *et al.* scheme is composed of the following elements:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) := (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; (m \cdot y^{k_0})^d, (g^{k_0})^d; (y^{k_1})^d, (g^{k_1})^d) \quad (3)$$

It corresponds to an ElGamal encryption of the message, and an ElGamal encryption of the element 1 (necessary to perform a key less re-encryption), along with an RSA [15] signature (exponentiation using d) of all these elements. To re-encrypt the ciphertext anyone can choose two values k'_0 and k'_1 , and perform the following operation:

$$(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1}; \alpha_2 \cdot \alpha_3^{k'_0}, \beta_2 \cdot \beta_3^{k'_0}; \alpha_3^{k'_1}, \beta_3^{k'_1}) \quad (4)$$

Klonowski *et al.* propose to accept the signature as valid if $\alpha_0 = \alpha_2^e$ holds, where e is the public verification key, corresponding to the signature key d (the RSA property is that $e \cdot d \pmod{(p-1)(q-1)} \equiv 1 \Rightarrow a^{e \cdot d} \pmod{N} \equiv a \pmod{N}$). This unfortunately does not guarantee that the ciphertext has not been modified, and does not therefore provide neither integrity nor non-repudiation as a signature scheme should.

2.1 Attacking the Scheme

The attack relies on the algebraic properties of RSA, in that the product of two signatures, results in the signature of the product, or more formally $m_0^d \cdot m_1^d = (m_0 \cdot m_1)^d$. Therefore if an attacker knows a signed plaintext m' , $(m')^d$, it can construct a valid Klonowski *et al.* ciphertext by multiplying it into another ciphertext in the following way:

$$(\alpha_0 \cdot m', \beta_0; \alpha_1, \beta_1; \alpha_2 \cdot (m')^d, \beta_2; \alpha_3, \beta_3) \quad (5)$$

The verification equation holds since $\alpha_0 \cdot m' = m \cdot m' \cdot y^x = (m \cdot y^k \cdot (m')^d)^e = (\alpha_2 \cdot (m')^d)^e$. The known plaintext and signature can therefore be multiplied into a valid ciphertext, at any stage, and produce another valid plaintext.

An adversary can also use two valid but unknown ciphertexts signed and encrypted to the same keys, and combine them to produce another valid, and apparently signed ciphertext.

$$(\alpha_0 \cdot \alpha'_0, \beta_0 \cdot \beta'_0; \alpha_1, \beta_1; \alpha_2 \cdot \alpha'_2, \beta_2 \cdot \beta'_2; \alpha_3, \beta_3) \quad (6)$$

Which would be a valid ciphertext since $m \cdot y_0^k \cdot m' \cdot y_0^{k'} = ((m \cdot y_0^k)^d \cdot (m' \cdot y_0^{k'})^d)^e$. Therefore an adversary can use ciphertexts, with unknown plaintexts and combine them into another valid ciphertext. This leads to existential forgery.

3 Breaking Onions Based on Universal Re-encryption

In Klonowski *et al.* [11] and Gomulkiewicz *et al.* [12] two very similar mix format schemes based on Universal Re-encryption are described. The first paper [11] discusses how such construction can be used to route messages in the mix network, including mechanisms for reply blocks and detours [4]. The second paper [12] claims that the use of Universal Re-encryption makes the mix scheme invulnerable to replay attacks. We will show that both schemes are vulnerable to tracing attacks by an adversary that observes the sender injecting an onion into the network, has the ability to use the network, and controls one corrupt mix.

The encoding schemes proposed are very simple. The sender (or a third party as described in [11]) wants to send a message m through a sequence of mixes $J_1, J_2, \dots, J_{\lambda+1}$, to the final receiver $J_{\lambda+1}$. The public keys corresponding to each node J_i are globally known and are $y_i = g^{x_i}$. Each address in sequence and the message is universally re-encrypted using El-Gamal:

$$\text{URE}_{x_1}(J_2), \text{URE}_{x_1+x_2}(J_3), \dots, \text{URE}_{x_1+x_2+\dots+x_\lambda}(J_{\lambda+1}), \text{URE}_{x_1+x_2+\dots+x_{\lambda+1}}(m) \quad (7)$$

$\text{URE}_x(m)$ denotes the ciphertext one gets by performing universal re-encryption on the message m under private key x . Note that only the public component $y = g^x$ of the private key x is required to perform this operation.

Routing and decryption are taking place in parallel. The onion is first relayed to J_1 , that uses its secret key x_1 to decode all blocks, retrieve J_2 and forward the message. There is no discussion in [11,12] about removing the blocks that have been decoded, or adding blocks to pad the message to a fixed size, but these can easily be done to hide the position of different mixes on the path and the overall path length.

3.1 Attacking the Scheme

Universal re-encryption, $\text{URE}_x(m)$, of a plaintext has some important properties that make our attacks possible. The ciphertext $\text{URE}_x(m)$ has two components: an ElGamal encryption of 1 under the public key g^x and the encryption of the message m under the same public key.

$$\text{URE}_x(m) \equiv (g_1^{k_1}, g_1^{k_1 x}, g_2^{k_2}, g_2^{k_2 x} m) \quad (8)$$

It is possible for anyone that knows $\text{URE}_x(m)$ to encrypt an arbitrary message m' under the same public key. Simply chose random k_3, k_4 and encode the message m' by multiplying it by the blinded encryption of 1:

$$\text{URE}_x(m') \equiv ((g_1^{k_1})^{k_3}, (g_1^{k_1 x})^{k_3}, (g_1^{k_1})^{k_4}, (g_1^{k_1 x})^{k_4} m') \quad (9)$$

Given $\text{URE}_x(m)$ it is easy to further encrypt it under an additional, arbitrary, key x_a and get $\text{URE}_{x+x_a}(m)$ without the knowledge of the secret x :

$$\text{URE}_{x+x_a}(m) \equiv (g_1^{k_1}, (g_1^{k_1})^{x_a} \cdot g_1^{k_1 x}, g_2^{k_2}, (g_2^{k_2})^{x_a} \cdot g_2^{k_2 x} m) \quad (10)$$

An interesting property is that $\text{URE}_x(m')$ is indistinguishable from $\text{URE}_x(m)$ by anyone who does not know the secret key x . Even if a party knows x it is impossible to determine that $\text{URE}_x(m')$ was derived from $\text{URE}_x(m)$.

We further note that each mix in fact acts as a decryption oracle:

1. The mix J_i receives an onion composed of universally re-encrypted blocks.

$$\dots, \text{URE}_{x_i}(J_{i+1}), \text{URE}_{x_i+x_{i+1}}(J_{i+2}), \dots, \text{URE}_{x_i+x_{i+1}+\dots+x_{\lambda+1}}(m) \quad (11)$$

2. The Mix J_i decrypts all blocks using its secret x_i . The result is:

$$\dots, J_{i+1}, \text{URE}_{x_{i+1}}(J_{i+2}), \dots, \text{URE}_{x_{i+1}+\dots+x_{\lambda+1}}(m) \quad (12)$$

3. The mix reads the next address J_{i+1} . If it is not well formed it stops (or starts the traitor tracing procedure described in Section 4.5 of [12]). Otherwise it re-encrypts all blocks and sends the resulting message to J_{i+1} .

Using the properties of universal re-encryption and the protocol that each mix implements an attacker that observes a message can trace it to its ultimate destination. Each block $\text{URE}_{x_1+\dots+x_i}(J_{i+1})$ is *replaced* by a block that redirects the onion to the corrupt node A followed by another block that contains the next address encrypted under the public key of the corrupt node x_a . A ‘label’ block that is the encryption of a fixed, per onion, label L has to also be included in order to be able to run multiple tracing attacks in parallel.

$$\begin{aligned} & \text{URE}_{x_1+\dots+x_i}(J_{i+1}) \\ & \leftarrow \text{URE}_{x_1+\dots+x_i}(A), \text{URE}_{x_1+\dots+x_i+x_a}(J_{i+1}), \text{URE}_{x_1+\dots+x_i+x_a}(L) \end{aligned} \quad (13)$$

Each mix J_i on the route will decode the message without realizing that it has been modified. Furthermore it will decode the block containing the address of the next mix J_{i+1} and the label L . The decoded message will contain:

$$\dots, A, \text{URE}_{x_a}(J_{i+1}), \text{URE}_{x_a}(L), \dots \quad (14)$$

The address A is interpreted by the honest mix J_i as the first address and the decoded message is redirected there. Once the adversary received it he can decode $\text{URE}_{x_a}(J_{i+1})$ and $\text{URE}_{x_a}(L)$ using his secret x_a to retrieve the next node J_{i+1} and the label L respectively.

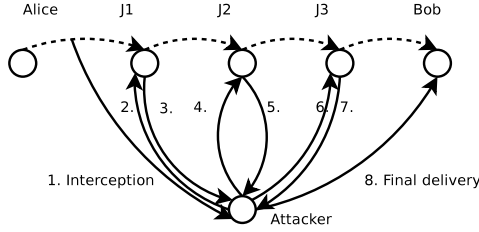


Fig. 1. After intercepting Alice’s mix packet, the attacker redirects the message to themselves

The attack results in the path of the traced onion becoming $J_1, A, J_2, A, J_3, A, \dots, A, J_{\lambda+1}$, as illustrated in Figure 1. The attacker is able to receive the onion every time it exists a mix, decode the next address and the label L , and re-insert it in the correct node to continue the tracing.

Our attack only requires a brief observation of the network to capture the onion to be traced. After that the onion is modified, and the mixes will not only decode the next address, but also forward that information to the attacker node. Therefore there is no need to perform any further passive or active attacks against messages in the network. Note that such onions can be traced even after they have been routed, since no duplicate detection mechanism is implemented. A replay prevention mechanism is difficult to implement in the context of universal re-encryption since all ciphertext (even of the same plaintext) are unlinkable without all the keys.

The fact that onions in a mix network are required to be of fixed size does not foil the attack. Since the linkage of the different parts of the message is so weak, it is possible to remove the tail blocks to allow for enough space to modify the message, as described above, to trace the connection. In case the message is too short to do this, it is still possible to perform the tracing in multiple steps, that only require replacing (over-writing) one section of the message to redirect it to the adversary. Then the same message is injected in the network with the next section / header overwritten to re-direct to the attacker again until the final recipient is found.

3.2 Replay and Tagging Attack

Besides the attack described above, the design in [12] fails to protect against replay attacks. An attacker can embed a tag that can be recognized after each mix J_i has processed the packet: he simply appends to or replaces the last block of the message with $\text{URE}_{\sum x_i + x_a}(L)$. Once the message is processed the output will contain $\text{URE}_{x_a}(L)$, which the adversary can decode to retrieve the label L . If the same message is inserted again it will output a message with the the same label, which leads to the classic replay attack.

Lu *et al.* [13] also point out that the scheme is susceptible to tagging attacks similar to those first proposed by Birgit Pfitzmann [16]. Their attack allows

a corrupt receiver to trace the message and uncover Alice as its sender. They correctly point out that this attack is outside the threat model of Klonowski *et al.* [11] and Gomulkiewicz *et al.* [12], since they assume that Alice and Bob trust each other. Our attacks do not make this assumption, and allow an arbitrary third party that acts as an active adversary and controls one node to fully trace and decrypt the messages exchanged.

4 Weaknesses of the rWonGoo Scheme

Lu *et al.* [13], note that Universal Re-encryption is susceptible to tagging attacks, but also propose rWonGoo, a novel anonymous communications scheme based on re-encryption. rWonGoo was designed to protect against tagging attacks, where an adversary modifies a message to trace it through the networks, and replay attacks, where a message is replayed to help tracing. We next provide a quick description of rWonGoo that will help us highlight its vulnerabilities (a full description is provided in [13]).

rWonGoo is broadly inspired by the Crowds anonymization scheme [14], and aims to be deployed in a decentralized network of thousands of peers. It assumes that an adversary is prevented from snooping on the network by link encryption, but may also control a fraction of nodes to assist the attack. The communication in rWonGoo is divided into two phases. In the first phase the channel is opened through the network between Alice and Bob, and the keys necessary to perform the re-encryption are distributed to all nodes through the channel. In the second phase messages between Alice and Bob can be exchanged. They start off being encrypted under the keys of all intermediary nodes, that each decrypt, re-encrypt and forward messages.

An rWonGoo channel is composed of two types of relaying nodes: those that perform re-encryption and those that are simply re-routing the message. The nodes that perform re-encryption, shall be called P_i (for $1 \leq i \leq \lambda$) with El-Gamal keys (g, y_i) respectively, while those that simply redirect shall be called Q_j (no keys are necessary since only redirection is taking place at nodes Q_j). Conceptually all communication between P nodes is done using a Crowds anonymous channel over Q nodes. In some sense rWonGoo routes already on top of a crowds anonymous channel. The final node P_λ is assumed to be Bob, the ultimate recipient of the anonymous messages from Alice (also P_0).

The channel establishment protocol is of special interest to an attacker. Alice first picks a node P_1 and extends her tunnel to it. This extension is done using the crowds protocol, until node P_1 is reached. The node P_1 sends back to Alice a set of potential next nodes, with their IP addresses, TCP ports and El-Gamal public keys. Alice chooses one of them and, through an encrypted channel described below, extends her tunnel to P_2 . The communications between P_1 and P_2 , are using the crowds protocol. This is repeated $\lambda - 1$ time until Alice instructs $P_{\lambda-1}$ to connect to Bob.

All communications between Alice and node P_i (including Bob i.e. P_λ) are encrypted in a layered manner. Alice always knows the public keys $y_1 \dots y_i$ and uses

them to generate a *key distribution* message that distributes to all intermediates $P_1 \dots P_i$ the keys necessary to re-encrypt messages. These are conceptually the composite public keys under which the messages seen by each P_i are encrypted. Alice sends the key distribution message:

$$A \rightarrow P_1 : ((y_1 \cdot \dots \cdot y_i)^r, g^r; y_0^{r'}, g^{r'}) \equiv (P_0^{\text{forward}}, P_0^{\text{backward}}) \quad (15)$$

P_1 removes his key from the first part of the message, to retrieve the public key $P_1^{\text{forward}} := ((y_1 \cdot \dots \cdot y_i)^r / (g^r)^{x_1}, g^r) \equiv (y_{1f}, g_{1f})$ necessary to re-encrypt messages traveling forward in the channel. Similarly he adds his public key to the second part of the message to calculate the key $P_1^{\text{backward}} := (y_0^{r'} \cdot (g^{r'})^{x_1}, g^{r'}) \equiv (y_{1b}, g_{1b})$ necessary to re-encrypt messages traveling back towards Alice. P_1 then sends the new key set $(P_1^{\text{forward}}, P_1^{\text{backward}})$ to node P_2 . This procedure is repeated by all P in the channel, until the final message arrives at P_i :

$$P_{i-1} \rightarrow P_i : (y_i^r, g^r; (y_0 \cdot \dots \cdot y_{i-1})^{r'}, g^{r'}) \quad (16)$$

This key distribution procedure ensures that all intermediate P_i know the public keys under which the messages they receive on the forward and backward path are encrypted. As a result they can decrypt them and re-encrypt them on their way. Upon receiving a message $M := (a, b)$ node P_j performs the decryption using its secret key (g, y_j, x_j) and re-encryption using the key $(g_{j(b|f)}, y_{j(b|f)})$, under which the message is encrypted, and passes the resulting M' to the next node in the path (using Crowds as transport).

$$M' := \text{ReEnc}_{(g_{j(b|f)}, y_{j(b|f)})}(\text{Dec}_{(g, y_j, x_j)}(M)) \quad (17)$$

Following this process a message sent from Alice to Bob encrypted under key P_0^{forward} , arrives encrypted under Bob's key (g, y_λ) , and a message send back from Bob to Alice under key $P_\lambda^{\text{backwards}}$ arrives encrypted under her key (g, y_0) .

4.1 Attacking rWonGoo: Capturing the Route

The key vulnerability of rWonGoo is that it is susceptible to man-in-the-middle attacks, that allow the rest of the channel to get captured after a malicious node is encountered. This means that after Alice chooses a bad node to include on the channel path, all subsequent nodes can be made to be bad too. The intuition behind this attack is that Alice knows very little about the network, and relies on intermediaries to discover other nodes and their public keys. She is therefore unable to tell the difference between a genuine interaction with the network, and a interaction that is simply simulated by an adversary.

The attacks proceeds quite simply: we assume that there is a first dishonest re-encrypting node on the path, named P_m . Once the dishonest node P_m receives the request to extend the channel, it starts simulating a network of nodes P_{m_k} , and provides Alice with their fictitious IP addresses, TCP ports and public keys (for which P_m knows the secret component). Alice chooses one of them to extend her tunnel, but no matter which one she chooses P_m never forwards any message but

keeps simulating more nodes, all running the rWonGoo protocol with Alice. Finally Alice connects to Bob, directly through P_m . Note that the fact that Alice is provided a choice of nodes to choose from does not eliminate any attacks, since they are all corrupt, or even non-existent. As Alice does not have any first hand experience of any of the nodes she is asked to choose (she cannot even query them to see if they exist, since this would reveal she is the originator of the tunnel), the attacker can populate these choices with not only malicious but also fictitious nodes.

During the key distribution phase of the protocol the malicious nodes substitute the keys communicated to Bob, for use in the backward channel, with their own keys. Therefore the key distribution message received by Bob is $(y_\lambda^r, g^r; y_m^r, g^r)$, where y_m is the public key of the adversary. As a result *any message sent by Bob back to Alice can be read by the malicious nodes*. Those messages can then be re-encrypted under the key f_m^b and sent to Alice.

Our attacks so far allows an adversary to perform a predecessor attack [17], and probabilistically find Alice after she engages in consecutive interactions with Bob. We can estimate how long, in terms of the number of fresh channels Alice has to open to Bob, the attack is likely to take. We assume that a fraction f of the network is controlled by the adversary [18]. The intersection attack succeeds immediately (for reasons explained below) if the first Crowds node after Alice, Q_1 belongs to the adversary, which it is with probability f . Consider the random variable L , which denoted the number of fresh rWonGoo channels that Alice opens to Bob, until a channel in which the first node Q_1 is corrupt. The random variable L follows a geometric distribution with parameter f , and Alice is on average expected to have $\mathbb{E}(L) = (1 - f)/f$ secure anonymous tunnels until her association with Bob is uncovered.

4.2 Decrypting Any Message Using Re-routing Oracles

First we note that any node in the network, including Alice and Bob, can be used as a *decryption oracle* for messages encrypted under their keys. During the key setup operation a node is asked to effectively decrypt the first part of the message it receives and relays it to the next node on the path. Consider the victim node P_i with public key y_i which is to be used to decrypt a ciphertext $m := (a, b) \equiv (g^k, y_i^k m')$. The adversary sets up an rWonGoo channel P_m, P_i, P'_m , where the nodes P_m and P'_m are controlled by the adversary. Then P_m sends to P_i the following message, that is to P_i indistinguishable from a key distribution message (k is a random factor chosen by the adversary):

$$P_m \rightarrow P_i : (b \cdot k, a; y_m^r, g^r) \quad (18)$$

The node P_i removes its key from the first component of the message and sends the result to the next node P'_m , which is also controlled by the adversary. The new message will be:

$$P_i \rightarrow P'_m : (b \cdot k / a^{x_i}, a; \dots) \quad (19)$$

As a result P'_m gets $b \cdot k / a^{x_i} = y_i^k m' \cdot k / y_i^k = m' \cdot k$ and can divide it by the known factor k to retrieve the encrypted message m' . We will denote the decryption of

a ciphertext m by the adversary as $m' = \text{Dec}_i(m)$, which only takes subscript i (and not the private key x_i) since it can be performed even if just the name of the node is known¹.

We have shown in the previous section that a malicious P_m can always uncover the receiver P_λ (or Bob) of any message seen, and see in clear all messages sent by Bob to Alice. Since any malicious node can also force any other node in the network to act as a decryption oracle, it follows that the attacker can also see in clear all messages sent by Alice to Bob. Each ciphertext m destined to Bob, has to travel through P_m , and is encrypted only under Bob's public key. The attacker can just use Bob as an oracle to retrieve the plaintext $m' = \text{Dec}_\lambda(m)$.

4.3 Using Any Q_m to Attack the Crowds Routing

In rWonGoo communication between any two P is done using the Crowds protocol, and we name the nodes that merely perform crowds redirection Q_i . Those simply forward the message and perform link encryption.

First, using the decryption attacks presented above, any corrupt Q_m node can capture the rest of the route until Alice asks to be connected to Bob. This is possible because the corrupt Q_m sees all the key distribution and actual messages that are relayed, starting from the first message in which Alice asks to have the rWonGoo channel connected to the next P_i on the route. At this point the corrupt rerouting node Q_m uses P_i as a decryption oracle to retrieve all information sent by Alice. As a result Q_m can simulate all interactions where the secret keys of P_i are needed, without ever relaying the channel through it. Our route capture attacks can now be performed by *any* corrupt P_m or just Q_m node on the path.

Secondly we note that a Q_m can test whether its predecessor is Alice by using it as a decryption oracle on a backwards message (which is only encrypted under Alice's key), and checking if the result is plaintext. In case the result is plaintext, Q_m can confirm that its predecessor is Alice. This turns the predecessor attack into an exact attack, and makes rWonGoo weaker than the original Crowds. Similarly the attacker can test any other node in the network to see if it is the originator of the message. This breaks anonymity after at most $\mathcal{O}(N)$ decryptions, where N is the size of the network, by a Q_m between Alice and P_1 .

A confirmation attack can be mounted by any Q_m , even if it is not on the Crowds route of the first hop between P_0 , or Alice, and the first mix P_1 . Any Q_m observes in clear the key $((y_0 \dots y_i)^{r'}, g^{r'}) \equiv m_s$, which is the combination of all the public keys of the P_i 's so far on the route. Q_m wants to test whether the path used is made of the guess set of nodes $P_{j_0} \dots P_{j_k}$. To do this Q_m can consecutively decrypt, using each of the nodes $P_{j_0} \dots P_{j_k}$ as oracles message

¹ Note that if the message decrypted using P_i as an oracle *is not* encrypted under the corresponding key y_i , it will result in a plaintext that is indistinguishable from random. This property can be used to detect valid decryptions, when the correct plaintext is expected to have some structure. In case the correct plaintext is also indistinguishable from random for the adversary, it is difficult to tell if the correct or incorrect node was used as a decryption oracle.

m_s , (i.e. $m'_s = \text{Dec}_{j_0}(\dots\text{Dec}_{j_k}(m_s))$). If it is the case that the plaintext equals one ($m'_s = 1$), then the guess is correct, and Q_m has established that the path so far was made of the nodes in the guess set. This is an all-or-nothing test that provides no partial information. As a result it does not scale well with the number of honest network nodes N and the path length l , since Q_m will have to perform $c = \binom{N}{l} \cdot l$ decryption requests.

4.4 The Stronger Crowds the Weaker rWonGoo

The complexity of the attack presented above, in terms of the parameters of the system, is counter intuitive. The attack becomes more difficult as the number of honest P s that re-encrypt the messages increases before the message is ‘seen’ by either a dishonest P_m or a dishonest Q_m (a node that only performs Crowds between P nodes, yet can see the ciphertext and perform the guessing attack). In case the message is seen by a corrupt Q_m as it is traveling between Alice and P_1 , only $\mathcal{O}(N)$ decryptions are required.

We assume that until a corrupt P_m or Q_m is reached, say node number v (at which point we can capture the route or perform the guessing attack) all nodes are selected uniformly at random. This allows us to calculate the expected position v of the first corrupt node, if we know that a certain fraction f of the network is corrupt. The number v follows a geometric distribution with parameter f and its expected value is $\mathbb{E}(v) = (1 - f)/f$. As the fraction of corrupt nodes increases we expect the message to be seen by the attacker earlier.

At the same time the Crowds protocol can be tuned with a parameter h , which is the probability a message is forwarded to its final destination (versus being forwarded to a random member of the crowd) by each node that receives it. It is also trivial to see that the average length u of each journey into the crowds subsystem (that is used to route between P s) follows a geometric distribution with parameter h , with average path length $\mathbb{E}[u] = (1 - h)/h$.

As mentioned before our guessing attack is most effective when the number of P s on the route is small, before the message is seen by the adversary. We know that on average the message will be seen in $\mathbb{E}(v) = (1 - f)/f$ hops, but the average length of its first Crowds trip between Alice (P_0) and the first re-encryptor P_1 is expected to be $\mathbb{E}[u] = (1 - h)/h$. We can conclude that if the parameter h is smaller than f (the corrupt fraction of nodes in the network) it is expected on average that the attacker will see the message on its first hop and be able to perform the most trivial guessing attack. The adversary only has to perform at most N decryption operations until Alice is revealed.

This result is counter-intuitive: the parameter h being smaller means that the number of intermediaries in the Crowds protocol is larger. One should expect this to increase the anonymity of the system. Contrary to this, increasing the length of the crowds path allows the adversary to observe the raw message earlier with higher probability, despite link encryption. Since the rWonGoo scheme is very vulnerable when the attacker can observe messages early on, increasing the ‘anonymity’ provided by the Crowds transport, decreases the overall anonymity of the system.

4.5 Partial Fix for rWonGoo

As it stands rWonGoo is weaker than Crowds (that only uses link encryption, and no other cryptographic protection.) It is possible to make its security as good as Crowds with a minor modification: the sender Alice, chooses a fresh key pair (g, y_0) for each channel. This would defeat the confirmation attacks that make rWonGoo weaker than crowds, since Alice cannot be forced to decrypt a ciphertext correctly, confirming that she is the sender. This makes rWonGoo as strong as Crowds, but much more complex and unnecessarily costly at the same time.

5 Conclusion

The properties of RSA that make the ‘Universal Re-encryption of signatures’ scheme vulnerable to our attacks have been known, and used in the past by Birgit Pfitzman to break anonymous communications schemes [16]. To overcome them special padding schemes such as PKCS#1 [19] are used to give ciphertexts a special structure that is infeasible to reconstruct by multiplying different ciphertexts together. These padding schemes require a verifier to have access to the message plaintext in order to verify its validity, making it therefore impossible to check the validity of re-encrypted ciphertexts (since they still hide the message m). To allow decrypted ciphertexts to be verified using a signature scheme none of the fancy cryptography is necessary: it is sufficient to encrypt using Golle *et al*, a signed message, and transmit the corresponding ciphertext. The receiver then decrypts the ciphertext and can check the signature. Therefore we see little hope in fixing this scheme while retaining its interesting re-encryption properties.

The attack against the onions based on universal re-encryption is possible because of many factors: We can modify the onions, since their integrity is not protected, and their different parts are not linked to each other in a robust manner. The mixes allow themselves not only to be used as decryption oracles for arbitrary ciphertexts, but also can be used to redirect traffic to the attacker making tracing effortless. Our attack shows that the claim in section 4.3 of [12], that the insertion of blocks in the onion structure is not possible, is unfounded which directly leads to our attack.

Finally we show that rWonGoo is a very fragile scheme. The additional cryptography in rWonGoo has made the overall system more susceptible to attack, than the original Crowds proposal, that only used link encryption. In particular it is possible for all messages between Alice and Bob to be read by the adversary with high probability, following route capture. Since any participant acts as a decryption oracle, it is possible to mount confirmation attacks to find Alice more quickly than if simple Crowds was used.

Our attacks lead to two important and novel intuitions, that anonymous communication system designers should carefully take into account in the future. First, the weakness of the rWonGoo scheme demonstrates that anonymous channels are not automatically composable: rWonGoo using the crowds protocols as a transport between mixes makes the system more vulnerable, not stronger.

Furthermore choosing more secure parameters for the Crowds transport used in rWonGoo, makes the overall scheme less secure, which is highly counter-intuitive.

Second, our attacks against the mix and signature schemes based on Universal Re-encryption, demonstrate the inherent difficulty in using this primitive in a secure fashion. Its power comes from its neat structure, which allows for re-encryption given only a ciphertext, and the use of multiple keys along with incremental decoding. It is these properties that made it a promising primitive for anonymous communications.

On the other hand to preserve these properties, and allow ciphertext to be universally re-encryptable, a designer is forced to let them be malleable, leak the public keys used, and is unable to add any redundancy for integrity checking of messages on their way. That is the weakness our attacks exploited, and it is a weakness that should have been foreseen given the rich literature on attacking re-encryption networks. The literature on (non-Universally) re-encryption networks demonstrates that, to be secured, such schemes require, identification of senders, expensive zero-knowledge proofs of knowledge of the plaintexts, and proofs of correct shuffle and threshold decryption. Such proofs have not yet been adapted to Universal Re-encryption, and would be difficult to adapt them to the dynamic setting of free-route mix networks, and the multiple threats that such networks face (dynamic membership, sybil attacks, . . .). Unless there is a breakthrough in this field, Universal Re-encryption will should always be used, in this context, with uttermost care.

Acknowledgements

Ross Anderson, Philippe Golle, Ari Juels have provided feedback that has improved this work. Discussions, over they years, with Paul Syverson about how to build mix networks based on Universal Re-encryption were invaluable to understand how to attack them. George Danezis is supported by the Cambridge-MIT Institute (CMI) project on ‘Third generation peer-to-peer networks’, the Flemish Research Council (FWO), and by the European Union PRIME project.

References

1. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
2. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003.
3. U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. Technical report, Network Working Group, May 25 2004. Internet-Draft.
4. Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium — NDSS ’96*, pages 2–16, San Diego, California, February 1996. IEEE.

5. Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In Tor Helleseth, editor, *Advances in Cryptology (Eurocrypt '93)*, volume 765 of *LNCS*, pages 248–259, Lofthus, Norway, 23-27 May 1993. Springer-Verlag.
6. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Pierangela Samarati, editor, *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 116–125. ACM Press, November 2001.
7. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.
8. Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's track*, San Francisco, USA, February 2004.
9. Peter Fairbrother. An improved construction for universal re-encryption. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 79–87. Springer, 2004.
10. Marek Klonowski, Mirosław Kutylowski, Anna Lauks, and Filip Zagorski. Universal re-encryption of signatures and controlling anonymous information flow. In *WARTACRYPT '04 Conference on Cryptology*, Bedlewo/Poznan, July 1–3 2004.
11. Marek Klonowski, Mirosław Kutylowski, and Filip Zagorski. Anonymous communication with on-line and off-line onion encoding. In Peter Vojts, Mria Bielikov, Bernadette Charron-Bost, and Ondrej Skora, editors, *SOFSEM 2005: Theory and Practice of Computer Science, 31st Conference on Current Trends in Theory and Practice of Computer Science*, Lecture Notes in Computer Science, pages 229–238, Liptovsk Jn, Slovakia, January 22–28 2005,. 3381.
12. Marcin Gomulkiewicz, Marek Klonowski, and Mirosław Kutylowski. Onions based on universal re-encryption – anonymous communication immune against repetitive attack. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications, 5th International Workshop, WISA 2004*, volume 3325 of *Lecture Notes in Computer Science*, pages 400–410, Jeju Island, Korea, August 23–25 2004. Springer.
13. Tianbo Lu, Binxing Fang, Yuzhong Sun, and Li Guo. Some remarks on universal re-encryption and a novel practical anonymous tunnel. In Xicheng Lu and Wei Zhao, editors, *ICCNMC*, volume 3619 of *Lecture Notes in Computer Science*, pages 853–862. Springer, 2005.
14. Michael K. Reiter and Aviél D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–38, 1999.
15. Ron L. Rivest, A. Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
16. Birgit Pfitzmann. Breaking efficient anonymous channel. In Alfredo De Santis, editor, *Advances in Cryptology (Eurocrypt '94)*, volume 950 of *LNCS*, pages 332–340, Perugia, Italy, 9-12 May 1994. Springer-Verlag.
17. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.*, 7(4):489–522, 2004.
18. John R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.
19. PKCS #1 v2.1: RSA Cryptography Standard. RSA Security Inc., June 2002.

Weak k -Anonymity: A Low-Distortion Model for Protecting Privacy

Maurizio Atzori^{1,2}

¹ CS Department, University of Pisa,
L.go B. Pontecorvo 3, 56127 Pisa, Italy

² KDD-Lab, ISTI-CNR,
V.G. Moruzzi 1, 56124 Pisa, Italy
atzori@di.unipi.it

Abstract. Sharing microdata tables is a primary concern in today information society. Privacy issues can be an obstacle to the free flow of such information. In recent years, disclosure control techniques have been developed to modify microdata tables in order to be anonymous. The k -anonymity framework has been widely adopted as a standard technique to remove links between public available identifiers (such as full names) and sensitive data contained in the shared tables. In this paper we give a *weaker* definition of k -anonymity, allowing lower distortion on the anonymized data. We show that, under the hypothesis in which the adversary is not sure a priori about the presence of a person in the table, the privacy properties of k -anonymity are respected also in the *weak k -anonymity* framework. Experiments on real-world data show that our approach outperforms k -anonymity in terms of distortion introduced in the released data by the algorithms to enforce anonymity.

Keywords: data privacy, k -anonymity, low distortion.

1 Introduction

Privacy protection can have different meanings and can be addressed at different levels. In this paper we concentrate on individual privacy, in the strict sense of *non-identifiability*, as prescribed by the European Union regulations on privacy, as well as US rules on protected health information (HIPAA rules). Privacy is regulated at the European level by Directive 95/46/EC (October 24, 1995) and Regulation (EC) No 45/2001 (December 18, 2000). In such documents, general statements about identifiability of an individual are given, such as:

“To determine whether a person is identifiable, account should be taken of all the means likely to be reasonably used either by the controller or by any person to identify the said person. The principles of protection should not apply to data rendered anonymous in such a way that the data subject is no longer identifiable.”

According to this perspective, we focus on *anonymity* of individuals, and analyze how the release of tables (databases) containing sensitive information may open up the risk of privacy breaches that may reveal individual identity.

An operative way to anonymize data is the removal of attributes that could potentially be used to identify specific individuals, based upon a set of guidelines (e.g., HIPAA rules.) Unfortunately, in some instances the practice of these guidelines does not achieve individual anonymity, while in other instances data are unnecessarily removed, resulting in the release of data that do not satisfy the information needs of the requesting parties. Violations in anonymity can occur when person-specific data records are uniquely joined with external tables containing *quasi-identifiers*. Fig. 1 illustrates this potential threat to privacy

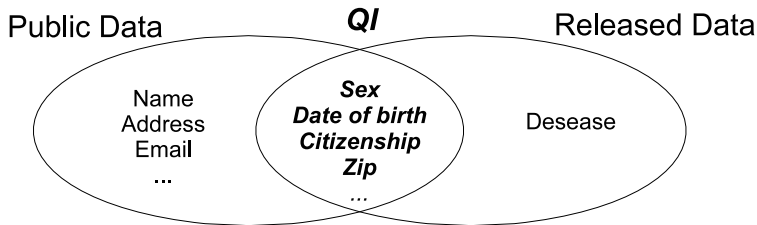


Fig. 1. Public data containing names and the released (private) data containing sensitive information such as diseases can be joined by using the quasi-identifiers QI

through the join of publicly available information (here referred to as *public data*) and medical claims records on a private table (*released data*) through a set of common attributes that can be used as quasi-identifiers (QI).

In literature, the concept of k -anonymity has been proposed to solve this problem, requiring each record to be indistinguishable from at least $(k - 1)$ other records with respect to a given set of attribute values in the released data [1, 2, 3, 4]. Although this definition of anonymity is simple, elegant and therefore extensively considered standard in literature, it has to face two practical problems when used in the context of real-world data (as we will point out in Section 2):

1. The running time requirements to find optimal anonymous tables are unpractical; and,
2. Even for optimal solutions, the distortion of the data introduced by the anonymization process is too high in terms of data quality, leading to unuseful tables.

To cope with this phenomenon in this paper we introduce an alternative to k -anonymity, namely *weak k -anonymity*, which overcomes both problems, since it requires to be enforced in just a subset of the records that do not respect k -anonymity. This property conduces to qualitatively better output results without giving up the privacy guarantees of original k -anonymity, allowing an effective use of non-optimal algorithms (i.e., also leading to reduced computational requirements.)

Paper Organization. The rest of this paper is organized as follows. Section 2 gives an overview of related work. In Section 3, we present our novel definition of weak k -anonymity and show its theoretical properties and relations with k -anonymity. In Section 4 we describe two algorithms to enforce weak k -anonymity on a given table. We point out the assumptions on the adversary knowledge in Section 5. Empirical results on the distortion introduced in the released tables are discussed in Section 6. Finally, Section 7 summarizes our research and concludes the paper.

2 Related Work

There are two general approaches to transform a set of partitioned records into a set of indistinguishable records [5,4]. One approach, referred to as *data generalization*, modifies similar records by replacing the distinct values of an attribute with a more general value (e.g., actual ages ranging from ‘21’ to ‘30’ replaced with the interval value ‘21-30’). The other approach referred to as *data suppression*, removes records from the table (tuple suppression), or merges some records into a single record by suppressing the individual values of one or more attributes (cell suppression). The objective is to generalize or suppress the minimum number of attribute values in order to satisfy a given k -anonymity requirement. Unfortunately, this optimization problem (even if relaxed to find approximated solutions within given bounds) is known to be NP-hard [6,7]. Several microaggregation techniques have been developed, including the **datafly** [4] and the **k-optimize** [8] algorithms. Although **k-optimize** has been proved to find optimal solution, as mentioned in Section 1, in practical cases two big problems arise: 1) the exponential time complexity makes it unfeasible if used with large datasets, and 2) the solution, despite its optimality, can be too much distorted to be useful [9].

In the following we present a relaxation of the k -anonymity problem, which we named *weak k -anonymity*, in order to overcome drawbacks of the (original) k -anonymity framework.

3 Weak k -Anonymity

Before defining *weak k -anonymity*, let us give some basic notation and review the original *k -anonymity* framework in a formal setting.

Definition 1 (Record). A record t is a vector in the form $(A_1 = v_{A_1}, \dots, A_n = v_{A_n})$. $\{A_1, \dots, A_n\}$ are called attributes of the record t . Given a set of attributes $A \subset \{A_1, \dots, A_n\}$, we say that $t[A]$ is the projection of the record t over the attributes A . $t[A]$ is the vector with $|A|$ elements obtained from t by removing the entries not in A .

Sometimes, we will also refer to records as rows or tuples.

Definition 2 (Tables). A table T is a multiset¹ of records with same attributes. Given a set of attributes $A \subset \{A_1, \dots, A_n\}$, we say that $T[A]$ is the projection of the table T over the attributes A . $T[A]$ is the multiset obtained by mapping each element $t \in T$ into $t[A]$.

From now on we will assume that information stored into records refers to specific individuals.

Definition 3 (k -Anonymity). A table T is k -anonymous w.r.t. a set of attributes QI if each record in $T[QI]$ appears at least $k - 1$ other times.

The idea behind this definition is that each record contains public available information in some attributes QI (often referred to as *quasi-identifiers*). The values of these attributes can be exploited to (almost uniquely) link those records to other records on other tables.

Definition 4 (Link). Given two records t_1 and t_2 we say that they are linked and use the notation $t_1 \rightleftharpoons t_2$ when in our domain they refer to the same entity.

In general, two records are linked if they have the same unique key. In relational databases, links between records of two tables are done by specifying the same value for the primary key. Notice that also sets of attributes can be used to uniquely link two records.

The goal of k -anonymity is therefore to prevent linking a record from a set of released records to a specific individual. As shown in Fig. 2, in this framework the adversary has access to all public available information (represented in the, possibly huge, *public table P*) that links names to other set of attributes (e.g., day

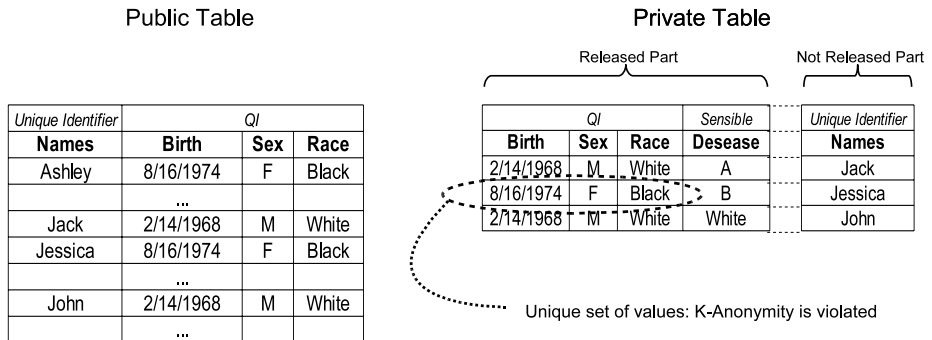


Fig. 2. A graphical representation of a violation of k -anonymity in the released part of the private table

¹ A set in which elements can appear more than once. Notice that, given a multiset M and an element e , we may have that $e \in M$ match more than once, i.e., $\{e \mid e \in M\}$ is a multiset and its cardinality can be larger than 1. The usual set operations are extended to multisets accordingly.

of birth, sex, race.) When a company (e.g., a medical institution) releases a table with sensitive data (*released table T*, i.e., the released part of the private table), then the adversary can play with quasi-identifiers (i.e., the attributes in common with the two tables) in order to discover unique links between records in the public and released tables. A k -anonymous table guarantees that the adversary cannot be able to link names to sensitive data by using the so-called quasi-identifiers, i.e., set of values that can be used as keys. The following property describes the desiderata behind k -anonymity.

Property 1 (Anonymity). An adversary cannot link the released data to less than k individuals by exploiting released and public data.

Intuitively, it means that, in the worst case, an adversary can only infer that a record is referred to one out of a certain k users. It is easy to prove that k -anonymity ensures that property.

Proposition 1. *Given a public identity information table P and a k -anonymous table T , Property 1 is satisfied.*

In other words, we can release a k -anonymous table without the risk of linking attacks. Now we have the necessary notation to give our novel definition of weak k -anonymity.

Definition 5 (Weak k -Anonymity). *Given an external public table P , we say that a table T is weakly k -anonymous, if*

$$\mathcal{P}(t_1 \rightleftharpoons t_2) \leq \frac{1}{k} \quad \forall t_1 \in P, t_2 \in T$$

The following proposition shows that k -anonymity implies weak k -anonymity, i.e., any k -anonymous table is also weakly k -anonymous.

Proposition 2. *Given a public table P and a k -anonymous table T , the following is satisfied:*

$$\mathcal{P}(t_1 \rightleftharpoons t_2) \leq \frac{1}{k} \quad \forall t_1 \in P, t_2 \in T$$

Proof. k -Anonymity requires that any linking information in T should appear at least other $k - 1$ times. Therefore, for each $t_1 \in P$, we have two cases: no $t_2 \in T$ which can be linked to t_1 , or, k or more $t_2 \in T$ that can be linked. In the first case the adversary can derive $\mathcal{P}(t_1 \neq t_2) = 1$, or equivalently $\mathcal{P}(t_1 \rightleftharpoons t_2) = 0$. Therefore, in the case at hand the inequality is satisfied. In the second case, the adversary cannot link t_1 to less than k records. Therefore, in terms of probability, since there are at least k choices with equal probability (records are indistinguishable), the adversary has no more than $\frac{1}{k}$ probability to guess the right link between two records. \square

Notice that the upper bound on the probability is tight, in the sense that if we use an upper bound smaller than $\frac{1}{k}$, the proposition is false. The above result shows that k -anonymity implies weak k -anonymity. The viceversa is false, since,

as we are going to show, we can have a valid weakly k -anonymous table with a row appearing less than k times. Although weak, Def. 5 surprisingly provides the same anonymity properties of the original k -anonymity under reasonable assumptions (see Section 5.)

Proposition 3. *Given a public identity information table P and a weakly k -anonymous table W , Property 1 is respected.*

Proof. It follows directly from the definition of weak k -anonymity. Since it requires that the probability of linking is no larger than $\frac{1}{k}$, the adversary has to choose among (at least) k different records. \square

One may wonder whether k -anonymity and weak k -anonymity are actually equivalent. Although apparently equivalent to k -anonymity, the definition of weak k -anonymity is strictly weaker. Infact, by focusing on Fig. 3 we can observe

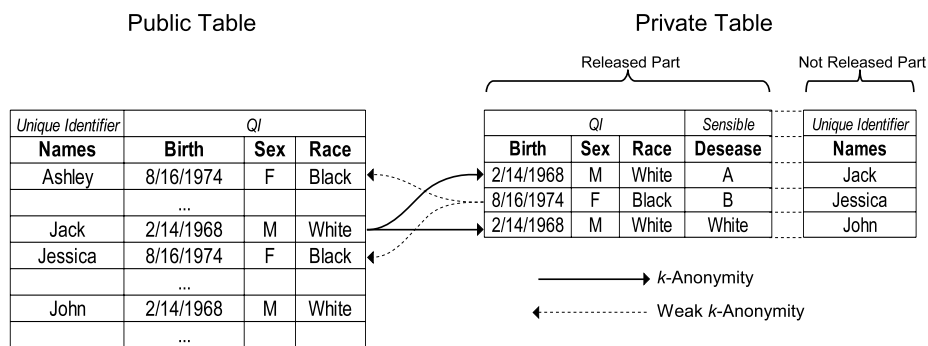


Fig. 3. A graphical representation of the adversary uncertainty exploited by k -anonymity and weak k -anonymity

how k -anonymity and weak k -anonymity blocks the adversary from linking information, thus guaranteeing a probability of linking less or equal to $\frac{1}{k}$. The continued arrows from left to right show that the first and the third records in the private table are 2-anonymous. In fact, because they are indistinguishable (values in QI are the same), the adversary cannot link ‘Jack’ to them (in the public table.) Notice that the second row *is not* 2-anonymous since values $Birth = 8/16/1974$, $Sex = F$, $Race = Black$ appear together in only one record.

The dashed arrows from right to left show instead that the second record in the private table is *weakly 2-anonymous*. In fact, although the record is not 2-anonymous, it can be linked to (at least) two individuals, ‘Ashley’ and ‘Jessica’. This means that the probability of linking the second tuple in the private table to any tuple in the public table is at most $\frac{1}{2}$. In other words, the private table is not 2-anonymous but it is weakly 2-anonymous. This is a good news, since as we will see later these differences allow weak k -anonymity to be less distortive while maintaing the same privacy guarantees.

4 Algorithms to Enforce Weak k -Anonymity

In this section we discuss how to exploit any existing algorithm for k -anonymity to release lower-distorted weak k -anonymity tables. Clearly, as shown by Prop. 2, any k -anonymity algorithm already releases weakly k -anonymity tables. But since we want to improve distortion, we would like to get weakly k -anonymous tables that are not necessarily k -anonymous.

The idea behind these algorithms is that, whenever the original algorithm verifies that a record occurs other $k - 1$ times in the released table, we instead verify that the record matches with at least k records in the public table. If it is so, we do not have to modify the record; otherwise, we enforce k -anonymity by tuple suppression or generalization.

We present **wabs** and **waboa**, two algorithms to enforce weak k -anonymity on tables. **wabs** (Algorithm 1.) is a very simple algorithm that tests records for weak k -anonymity on the set of records in the public information table (line 5). It suppresses (i.e., does not allow to release) records in the private table that can be linked to less than k records in the public table. The time complexity of **wabs** is trivially linear if the public table P is hash-indexed on QI , $O(|P| \log |P|)$ otherwise (since we can fairly assume $|P| \geq |T|$).

Algorithm 1. Wabs - Weak Anonymity By Suppression

Input: public available table P ; private table T ;

Output: weakly k -anonymous table W

- 1: $QI \leftarrow \text{Attributes}(P) \cap \text{Attributes}(T)$
 - 2: $W \leftarrow \emptyset$; // table W will contain weakly k -anonymous records
 - 3: **for all** $t_T \in T$ **do**
 - 4: $S \leftarrow \{t_P \in P \mid t_P[QI] \text{ matches } t_T[QI]\}$; // S is the "link table" of t_T
 - 5: **if** $|S| \geq k$ **then**
 - 6: $W \leftarrow W \cup \{t_T\}$;
 - 7: **output** W ;
-

The second algorithm, **waboa** (Algorithm 2.), exploits any existing algorithm for k -anonymity (line 2) to enforce the k -anonymity requirement on the public table. Then, the algorithm substitutes the actual values in the private table with the corresponding generalized or suppressed values (depending on the k -anonymity algorithm used) in the anonymized table (line 8.)

Since, in general, it is expected that the public table has more records than the private table to be released, the k -anonymization will be less distortive. For **waboa**, time complexity is again $O(|P| \log |P|)$ if the public table is not indexed on QI , plus the complexity of the algorithm for k -anonymity.

It is worth to note that both algorithms described so far *do not* ensure k -anonymity but instead they enforce only weak k -anonymity. This fact leads to lower distortion of the resulting anonymized table if compared with k -anonymity algorithms, as shown in Section 6.

Algorithm 2. Waboa - Weak Anonymity By Outer Anonymization

Input: public available table P ; private table T ;**Output:** weakly k -anonymous table W

```

1:  $QI \leftarrow \text{Attributes}(P) \cap \text{Attributes}(T)$ 
2:  $P^k \leftarrow k\text{-anonymity-algorithm}(P, QI)$ ; //  $P^k$  is anonymized on  $QI$ 
3:  $W \leftarrow \emptyset$ ; // table  $W$  will contain weakly  $k$ -anonymous records
4: for all  $t_T \in T$  do
5:    $S \leftarrow \{t_{P^k} \in P^k \mid t_{P^k}[QI] \text{ matches } t_T[QI]\}$ ; //  $S$  is the "link table" of  $t_T$ 
6:   // ASSERT:  $|S| \geq k$ , since  $P^k$  is  $k$ -anonymous on  $QI$ 
7:    $t_S \leftarrow$  an element from  $S$  (any);
8:    $W \leftarrow W \cup \{t_T[\text{Attributes}(T) \setminus QI] \cup t_S[QI]\}$ ;
9: output  $W$ ;
```

5 Discussion on Weak k -Anonymity

In this section we point out the differences between weak k -anonymity and k -anonymity, and discuss about what assumptions must be true in order to have the *same* anonymity guarantees but with lower distortion. We showed so far that weak k -anonymity avoids linking attacks as well as the k -anonymity. This is true under two assumptions:

1. the table P containing public information must be available to the data owner;
2. the adversary must not know a priori (i.e., before accessing the released table) whether an entity is in the table or not.

The former assumption is slightly stronger than the one used in the (non-weak) k -anonymity framework. Here, instead of needing just the set of quasi-identifier, we require the public table. Notice that also a subset of the information contained in the public table can improve the final distortion of the anonymized table. This requirement needs to be respected in order to minimize the distortion as much as possible. However, there is no risk of loss of privacy if it is not met. Infact, in the very worst case in which the publisher has no access to the public table, k -anonymity and weak k -anonymity collapse becoming completely equivalent.

The latter is also quite reasonable, but there are applications in which it does not hold. Whenever the adversary is a priori informed about the presence of some entities in the released table, then weak k -anonymity can be faulty in anonymity preservation. Although this kind of attack also affects k -anonymity (see d -relative anonymity in [10] and ℓ -diversity in [11]), in the context of weak k -anonymity this could be more severe.

Example 1. Suppose it is publicly known that in our district there are 947 people s.t. $age = 53$, $sex = M$, $education = Ph.D$. Since in our private table (that we are willing to share) there is only one person matching such characteristics, he is 1-anonymous and weakly 947-anonymous. But releasing information such $age = 53$, $sex = M$, $education = Ph.D$ and $disease = A$ (allowed by the weak k -anonymity if $k \leq 947$) is not secure if the adversary already knows that the

person matching $name = Jack$, $age = 53$, $sex = M$, $education = Ph.D$ is for sure in the private database, since it is possible to derive $\mathcal{P}(name = Jack \Leftrightarrow disease = A) = 1$.

Since in several applications k -anonymity is impractical because of the high distortion [9], we suggest an hybrid approach when the adversary is expected to have information on the presence of an entity in the private table: releasing weakly k -anonymous tables such that are also l -anonymous with $l \ll k$. In fact, we can prevent such kind of attacks (as the one in Example 1) by assuring a minimum level of k -anonymity in the released data, without compromising the output quality.

6 Experiments on Real-World Data

In this section we experimentally prove that weak k -anonymity is feasible and sensibly less distortional than k -anonymity. The results confirm that, even with the naïve method of suppression, enforcing weak k -anonymity is feasible for real-world applications, and outperforms k -anonymity in terms of number of suppressed tuples.

For the experiments, as a public table we used the well-known `census-income` dataset [12], containing census, demographic and employment data extracted from the 1994 and 1995 current population surveys conducted by the US Census Bureau, with 41 attributes and 299285 records². For the private table, we used a subset of the records in the public table (1% to 5% of the public table, as expected in the real application scenarios.) Therefore we suppose that the adversary has access to the `census-income` dataset (without the sensitive data) and that we have a smaller table containing private information to be kept anonymous. The quasi-identifiers used in the experiments are: age, education, marital status, major occupation code, sex, country of birth, citizenship.

Experiment 1. We compared k -anonymity and weak k -anonymity in terms of percentage of records in the released dataset (1% or 5% of the whole set) that need to be suppressed in order to accomplish k -anonymity and weak k -anonymity requirements.

The experimental results are shown in Fig. 4. Weak k -anonymity clearly outperforms k -anonymity. For instance, consider we want to release the table being sure that the adversary can guess the right link of released records on our table with public records with probability at most 10%, i.e., $k = 10$. In this case we have that 73.6% of the tuples are not k -anonymous (and therefore need to be suppressed or generalized, depending on the algorithm), while only 30.8% are not weak k -anonymous.

It is also worth noting that weak k -anonymity is very scalable on the k parameter, while k -anonymity has a peak around $k = 150$ because no record in the randomly-chosen 5% of the dataset was 150-anonymous³. Another important

² The dataset was actually split into training data and test data for classification purposes. In our study we merged them back.

³ Infact, the most frequent record was found 138 times in the dataset.

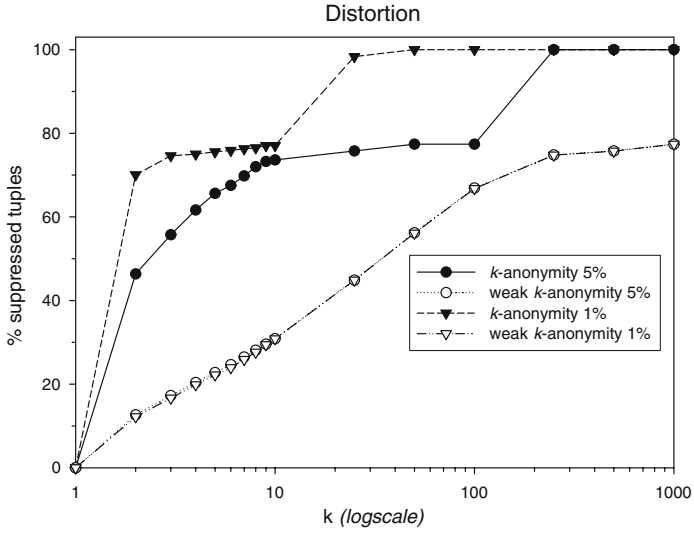


Fig. 4. The graph shows the percentage of records whose k -anonymity and weak k -anonymity are violated while varying the privacy threshold k

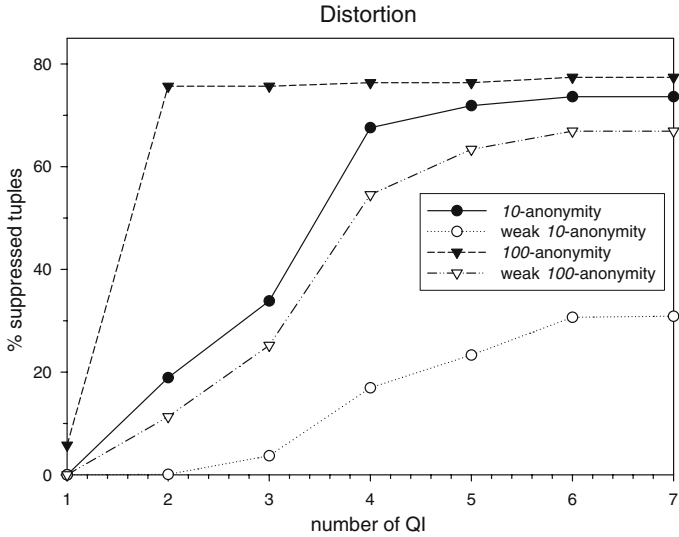


Fig. 5. The graph shows the percentage of records whose k -anonymity and weak k -anonymity are violated joining while varying the number of QI used

remark is that weak k -anonymity distortion does not sensibly depend on the size of the subset we use, while k -anonymity became unpractical for small tables.

Experiment 2. In this experiment we compare k -anonymity and weak k -anonymity by using a 5% private table and varying the number of quasi-identifiers.

The results are shown in Fig. 5. Weak k -anonymity always outperforms k -anonymity. For example, weak 100-anonymity, that guarantees $\mathcal{P} = 1\%$ of linking attacks, needs to suppress less records than 10-anonymity, that ensures only $\mathcal{P} = 10\%$.

7 Conclusions

In this paper we gave a *weaker* definition of k -anonymity, which allows lower distortion when enforcing anonymity on the releasing tables. We showed that, under the hypothesis in which the adversary cannot be sure a priori about the presence of a person in the table, the privacy properties of k -anonymity are respected also in the *weak k -anonymity* framework. Experiments on real-world data showed that our approach outperforms k -anonymity in terms of distortion introduced in the released data by the algorithms to enforce anonymity.

Acknowledgments. The author wishes to thank Federica Paci, Christopher W. Clifton, Wei Jiang and Mehmet Ercan Nergiz for their comments on an early version of this paper; Fosca Giannotti, Dino Pedreschi and Franco Turini for their encouragement, support and opportunity to work on data privacy problems.

*This work has been funded by the EU project GeoPKDD (IST-6FP-014915).*⁴

References

1. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (abstract). In: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems (PODS), New York, NY, USA, ACM Press (1998) 188
2. Sweeney, L.: k -Anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(5) (2002) 557–570
3. Samarati, P.: Protecting respondents’ identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* **13**(6) (2001) 1010–1027
4. Sweeney, L.: Achieving k -anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(5) (2002) 571–588
5. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* **14**(1) (2002) 189–201

⁴ More information available at <http://www.geopkdd.eu/>

6. Meyerson, A., Williams, R.: On the complexity of optimal k -anonymity. In: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS). (2004) 223–228
7. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k -anonymity. *Journal of Privacy Technology* **1**(2005112001) (2005)
8. Bayardo, R., Agrawal, R.: Data privacy through optimal k -anonymization. In: Proceedings of the 21st International Conference on Data Engineering (ICDE), Washington, DC, USA (2005) 217–228
9. Aggarwal, C.C.: On k -anonymity and the curse of dimensionality. In: Proceedings of the 31st International Conference on Very Large Databases (VLDB), Trondheim, Norway, VLDB Endowment (2005)
10. Øhrn, A., Ohno-Machado, L.: Using boolean reasoning to anonymize databases. *Artificial Intelligence in Medicine* **15**(3) (1999) 235–254
11. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l -Diversity: Privacy beyond k -anonymity. In: Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE), Atlanta, GA, USA (2006)
12. Hettich, S., Bay, S.: The UCI KDD Archive. <http://kdd.ics.uci.edu/> (1999)

Protecting Data Privacy Through Hard-to-Reverse Negative Databases

Fernando Esponda¹, Elena S. Ackley², Paul Helman²,
Haixia Jia², and Stephanie Forrest²

¹ Department of Computer Science
Yale University
New Haven, CT 06520-8285
fesponda@cs.yale.edu

² Department of Computer Science
University of New Mexico
Albuquerque, NM 87131-1386

{hjia, elenas, forrest, helman}@cs.unm.edu

Abstract. The paper extends the idea of negative representations of information for enhancing privacy. Simply put, a set DB of data elements can be represented in terms of its complement set. That is, all the elements not in DB are depicted and DB itself is not explicitly stored.

We review the negative database (NDB) representation scheme for storing a negative image compactly and propose a design for depicting a multiple record DB using a collection of NDB s—in contrast to the single NDB approach of previous work. Finally, we present a method for creating negative databases that are hard to reverse in practice, i.e., from which it is hard to obtain DB , by adapting a technique for generating 3-SAT formulas.

1 Introduction

Protecting sensitive data—controlling access to information and restricting the types of inferences that can be drawn from it—is a concern that has to be continually addressed in a world where the demands on the availability of data, and the criteria for its confidentiality evolve. Current technologies of encryption (for the data itself) and query restriction (for controlling access to data) help ensure confidentiality, but neither solution is appropriate for all applications. In the case of encryption, the ability to search data records is hindered; in the case of query restriction, individual records are vulnerable to insider attacks. Furthermore, many of the current solutions rely on the same set of assumptions, e.g., prime factoring—diversification in approaches ensures robustness.

In this paper, we discuss an approach for representing data that addresses some of these concerns and provides a starting point for the design of new applications. A motivating scenario involves a database of personal records which an outside entity might need to consult, for example, to verify an entry in a watch-list. It is desirable to have a database that supports a restricted type of

queries, disallowing arbitrary inspections (even from an insider), and that can be updated without revealing the nature of the changes to an onlooker.

In our approach, the negative image of a set of data elements is represented rather than the elements themselves (Fig. 1). Initially, we assume a universe U of finite-length strings (or records), all of the same length l and defined over a binary alphabet. We logically divide the space of possible strings into two disjoint sets: DB representing a set of positive records (holding the information of interest), and $U - DB$ denoting the set of all strings not in DB . We assume that DB is uncompressed (each record is represented explicitly), but we allow $U - DB$ to be stored in a compressed form called NDB . We refer to DB as the *positive database* and NDB as the *negative database*. From a logical point of view, either will suffice to answer questions regarding DB ; however, they present different advantages. For instance, in a positive database, inspection of a single string provides meaningful information; inspection of a single ‘negative’ string reveals little about the contents of the original database. Given that the positive tuples are never stored explicitly, a negative database could be much more difficult to misuse.

The basic concept was introduced in [19,16] and establishes the general theoretical foundations for some of the properties of the representation, especially with regards to privacy and security. The present goal is to address some of the practical concerns regarding the security of negative databases, and the efficiency of updating them. We introduce a novel storage design that better supports update operations and adapt techniques from other fields to create negative databases that are more secure in practice.

In the following section we review the negative database representation, give some examples, and explain how to query it. We then investigate some of the implications the approach has for privacy and security. In particular, we take advantage of the fact that the general problem of recovering the positive set from our negative representation is \mathcal{NP} -hard (see Ref. [19,20,16]) and present a novel scheme that creates negative databases that are indeed hard to reverse. We introduce a setup that overcomes some of the update inefficiencies of previous approaches and, finally, review related work, discuss the potential consequences of our results, and outline areas of future investigation.

2 Representation

In order to create a negative database (NDB) that is reasonable in size, we must compress the information contained in $U-DB$ while retaining the ability to answer queries. We introduce an additional symbol to the binary alphabet, known as a “don’t-care” and written as $*$. The entries in NDB will thus be strings of length l over the alphabet $\{0, 1, *\}$. The don’t-care symbol has the usual interpretation and represents a one and a zero at the string position where the $*$ appears—positions set to one or zero are referred to as “defined positions”. The use of this new symbol allows for large subsets of $U - DB$ to be depicted with just a few entries in NDB (see example in Fig. 1).

A string s is taken to be in DB if and only if s fails to match all the entries in NDB . The condition is fulfilled only if for every string $y \in NDB$, s disagrees with y in at least one defined position.

$DB \ U - DB \ NDB$		
000	001	001
100	010	*1*
101	011	
	110	
	111	

DB	$U - DB$	NDB
0001	0000	11**
0100	0010	001*
1000	0011	011*
1011	0101	0000
	0110	0101
	0111	1001
	1001	1010
	1010	
	1100	
	1101	
	1110	
	1111	

Fig. 1. (a),(b). Different examples of a DB , its corresponding $U-DB$, and a possible NDB representing $U-DB$.

Boolean Formula	NDB
$(x_1 \text{ or } x_2 \text{ or } \bar{x}_5) \text{ and}$	00**1
$(\bar{x}_2 \text{ or } x_3 \text{ or } x_5) \text{ and}$	*10*0
$(x_2 \text{ or } \bar{x}_4 \text{ or } \bar{x}_5) \text{ and } \Rightarrow$	*0*11
$(\bar{x}_1 \text{ or } \bar{x}_3 \text{ or } x_4)$	1*10*

Fig. 2. Mapping SAT to NDB : In this example the boolean formula is written in conjunctive normal form (CNF) and is defined over five variables $\{x_1, x_2, x_3, x_4, x_5\}$. The formula is mapped to a NDB where each clause corresponds to a record, and each variable in the clause is represented as a 1 if it appears negated, as a 0 if it appears un-negated, and as a * if it does not appear in the clause at all. It is easy to see that a satisfying assignment of the formula such as $\{x_1 = \text{FALSE}, x_2 = \text{TRUE}, x_3 = \text{TRUE}, x_4 = \text{FALSE}, x_5 = \text{FALSE}\}$ corresponding to string 01100 is *not* represented in NDB and is therefore a member of DB .

Queries are also expressed as strings over the same alphabet; when a string, Q , consists entirely of defined positions—only zeros and ones—it is interpreted as “Is Q in DB ?”, and we refer to it as a simple membership or authentication query. Answering such a query requires examining NDB for a match, as described above, and can be done in time proportional to $|NDB|$. On the other hand, the work in [19] demonstrates an efficient mapping between boolean satisfiability formulas and NDB s (see Fig. 2) and shows that the problem of reversing a NDB —recovering DB —is \mathcal{NP} -hard even when the size of the resulting DB is polynomial in the input size—determining the size of DB or even if it’s empty

or not is \mathcal{NP} -hard as well. Consequently, answering queries with an arbitrary number of $*$ symbols is also intractable.

Take, for example, a negative database of the tuples $\langle \text{name, address, profession} \rangle$. The query “Is $\langle \text{Tintan, 69 Pine Street, Plumber} \rangle$ in DB ?” (written as a binary string Q) would be easily answered, while retrieving the names and addresses of all the engineers in DB (expressed as a query string with the profession field set to the binary encoding of ‘engineer’ and the remaining positions to $*$) would be intractable. Note that it is possible to construct $NDBs$, with specific structures, for which complex queries can be answered efficiently (see Refs. [19,16]). Indeed, creating negative databases that are hard to reverse in practice can be a difficult task; in the next section, we address this issue and present an algorithm for creating negative databases that only support authentication queries efficiently.

3 Hard-to-Reverse Negative Databases

The creation of negative databases has been previously addressed in [19,17,20,16], where several algorithms are given that either produce $NDBs$ that are provably easy to reverse, i.e., for which there is an efficient method to recover DB , or that have the flexibility to produce hard-to-reverse instances in theory, but have yet to produce them experimentally. It was shown in [19] that reversing a NDB is an \mathcal{NP} -hard problem, but this, being a worst case property, presents the challenge of creating hard instances in practice.

In this section, we focus on a generation algorithm that aims at creating hard-to-reverse negative databases in practice; we take advantage of the relationship negative databases have with the boolean satisfiability problem (SAT) (Fig. 2) and look into the body of work devoted to creating difficult SAT instances (e.g., [39,2,31,30]). As an example, we focus on the model introduced in [30] and use it as a basis for creating $NDBs$. The resulting scheme has two important differences with the algorithms of Refs. [19,17,20,16] besides the ability to produce hard instances: first, it generates an NDB for each string in DB , and second, it creates an inexact representation of $U-DB$, meaning that some strings in addition to DB will not be matched by NDB .

In what follows we present the generation algorithm, outline how the problem of extra strings can be dealt with, and empirically show that the resulting databases are hard to reverse.

3.1 Using SAT Formulas as a Model for Negative Databases

Reference [30] presents an algorithm for creating SAT formulas which we use as the basis for our negative database construction. Their objective is to create a formula that is known to be satisfiable, but which SAT-solvers are unable to settle. The approach is to take an assignment A (a binary string representing the truth values for the variables in the formula), and create a formula satisfied by it—much like the algorithms in [19,17,20,16], except that the resulting formula

might be satisfied by other *unknown* assignments. Given the assignment A , the algorithm randomly generates clauses with $t > 0$ literals satisfied by it with probability proportional to q^t for $q < 1$ (q is an algorithm specific parameter used to bias the distribution of clauses within the formula). The purpose of the method is to balance the distribution of literals in such a way as to make formulas indistinguishable from one another in this respect. The process outputs a collection of clauses, all satisfied by A , which can be readily transformed into a negative database (see Fig. 2).

Given a database (DB) of size at most one (Sect. 3.4 discusses DB s with more than one record), containing a l -length binary string A , we create a negative database (NDB) with the following properties:

1. Each entry in the negative database has exactly three specified bits.
2. A is not matched by any of NDB 's entries.
3. Given an arbitrary l -bit string, it is easy to verify if it belongs to NDB or not (in time proportional to the size of NDB).
4. The size of NDB is linear in terms of the length of A . Let l be the number of bits in A and m the number of strings in NDB ; the tunable parameter $r = m/l$ determines the size of the database and its reversal difficulty.
5. The size of NDB does not depend on the contents of DB , i.e., it has the same size for $|DB| = 1$ and $|DB| = 0$.
6. A is “almost” the “only” string not matched by NDB , i.e., almost the only string contained in the positive image DB' of NDB . The other entries in DB' are close in hamming distance to A (see Sect. 4).
7. The negative database NDB is very hard to reverse, meaning no known search method can discover A in a reasonable amount of time (provided that the number of bits in A be greater than 1000, as explained below).

Properties one through five follow from the isomorphism of negative databases with a 3-SAT formulae (see Fig. 2) and the characteristics of the algorithm. Point six is addressed in the next section, and completes the negative database generation scheme. Property seven is ascertained empirically in Sect. 3.3.

3.2 Superfluous Strings

A consequence of the above method for generating negative databases, is the inclusion of extra strings in the corresponding positive database. That is, DB' —the reverse of NDB —will include strings that are not in the original DB from which it was created; we refer to these strings as superfluous¹.

Figure 3 displays the expected number of strings not represented by NDB (and hence members of DB') as a function of their normalized Hamming distance to A —the true member of DB —and shows that all superfluous strings are within 0.13 distance from A (for the given parameter settings)².

¹ Note that $DB \subseteq DB'$.

² The definition of the plotted function is: $f(\alpha) = \frac{1}{\alpha^\alpha(1-\alpha)^{1-\alpha}} \left(1 - \frac{(q(1-\alpha)+\alpha)^k - \alpha^k}{(1+q)^{k-1}} \right)^r$, for details see [30].

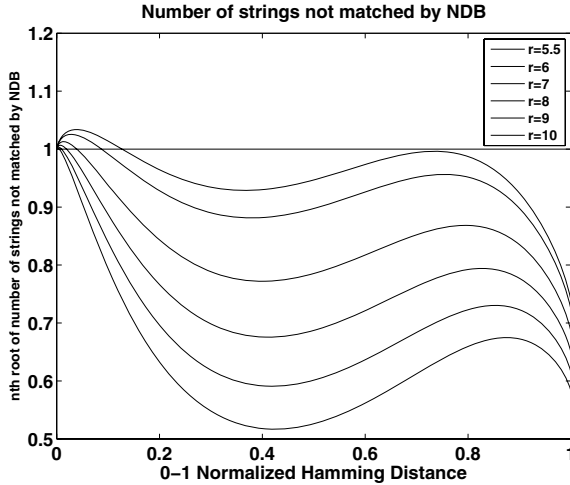


Fig. 3. Number of strings not matched by NDB (members of DB') as a function of the hamming distance to A —the original DB entry. The plot shows the expected numbers for $q = 0.5$ and several r values: from top to bottom $r = 5.5, 6.0, 7.0, 8.0, 9.0, 10.0$. An interplay between q and r determines how difficult the NDB will be to reverse and how many “extra” strings will go unmatched by NDB .

Increasing the value of r reduces the number of superfluous strings; however, it also increases the size of the database and, more importantly, leads to NDB s that are potentially easier to reverse (see [25,3,1]).

To address the incidence of superfluous strings, we introduce a scheme that allows us to distinguish, with high probability, the true members of DB from the artifacts. Rather than creating a NDB using A as input, we construct a surrogate string A' —appending to A the output of some function F of A —and use it to generate NDB . The membership of an arbitrary string B is established by computing $F(B)$ and testing whether B concatenated with $F(B)$ is represented in NDB ³. The purpose of the function is to divide the possible DB' entries into valid and invalid—valid strings having the correct output of F appended to them—and reduce the probability of including any unwanted valid strings in DB' .

The choice of function impacts both the accuracy of recovery (avoidance of superfluous strings) and the performance of the database: the more bits appended to A , the less likely to mistake a false string for a true one (assuming a reasonable code) and the larger the resulting NDB . There is a wide variety of codes that can be used for this purpose: parity bits, checksums, CRC codes, and even hash functions like SHA or MD5 with upwards of a 100 bits⁴.

³ Naturally F needs to be public known.

⁴ It's important to emphasize that the proposed scheme relies on F solely for reducing the incidence of false entries and not, in any way, for the secrecy of the true ones.

To provide an idea of how the function impacts accuracy we consider a general model which assumes, for simplicity, valid strings are uniformly distributed and sampling with replacement. The chance of randomly finding a valid string is 2^{-c} , where c is the number of bits introduced by the function. The probability of including an unwanted valid string is $1 - (1 - 2^{-c})^{|DB'|}$, where $|DB'|$ is the number of strings unmatched by NDB . The model illustrates (see Fig. 4) the dependence of accuracy on the code size—the density of valid strings—and the number of strings introduced by the generation algorithm. Clearly, a sophisticated code such as the CRC, which attempts to maximize the minimum hamming distance between valid strings, will greatly increase the accuracy of section’s 3.1 generation scheme.

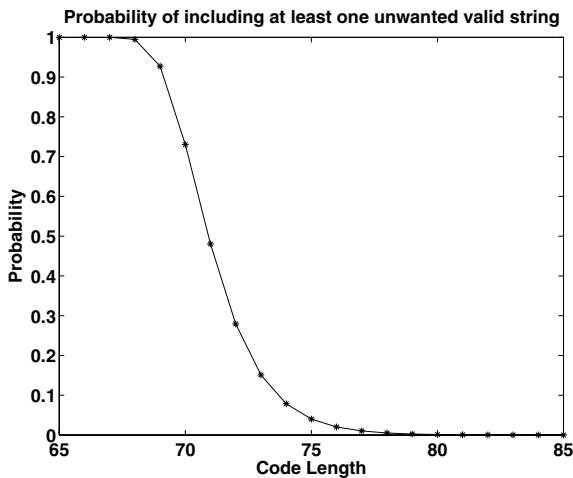


Fig. 4. Probability of including an unwanted valid string as a function of the error correcting code, c , according to $1 - (1 - 2^{-c})^{|DB'|}$. $|DB'|$ denotes the expected number of strings unmatched by NDB ; it is calculated for a string length, l , of 1000 and $r = 5.5$.

3.3 Hardness

To illustrate how hard to reverse these NDB s are, we produced instances for strings ranging from 50 to 300 bits in length and $r = 5.5$. Their difficulty is assessed by the ability of well established SAT-solvers to find a string in DB' . There are two types of solvers: complete and incomplete. Complete solvers search the space exhaustively, while incomplete solvers explore only a fraction of it and can handle much larger instances (in terms of string length l); however, unlike complete solvers, their failure to find a solution does not imply that one doesn't exist.

Figure 5 shows the results for the zChaff complete solver (zChaff is often the champion of the yearly SAT competition) and Fig. 6 shows the results for WalkSAT, a well known incomplete solver. The experiments show that both zChaff and WalkSAT find a DB' entry in time exponential in the length of the

string l . Consider that fully reversing NDB , i.e., finding all of the strings in DB' , will entail running the solver $|DB'| + 1$ times (the extra run is to establish that there are no more strings left). Additionally, we tested 100 NDB s with $l = 1000$ on zChaff and WalkSAT, as well as on two other solvers: SATz and SP (the first complete the second incomplete). No DB' entry was found for any of them before the incomplete solvers terminated and the complete solvers ran out of memory.

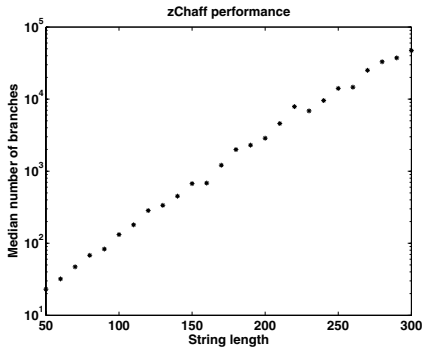


Fig. 5. Running time of zChaff on NDB with strings of length l , ranging from 50 to 300

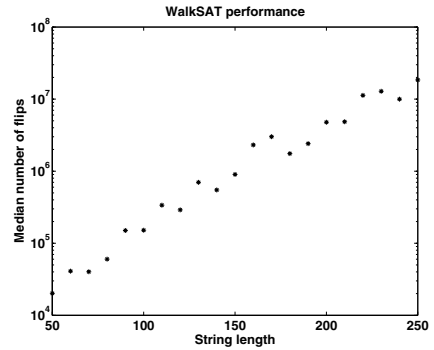


Fig. 6. Running time of WalkSAT on NDB with strings of length l , ranging from 50 to 300

3.4 Multi-record Negative Databases

The preceding section explored how to create a hard-to-reverse negative representation of a DB with zero or one entries; now, we briefly outline how this can be extended for DB s of an arbitrary size—the work in [19,17] is concerned with creating negative databases for any DB , regardless of its size, but does not show that the instances they output are hard to reverse in practice.

Our scheme can be used to generate the negative representation of any set of strings DB by creating an individual NDB_{A_i} for each string A_i in DB , i.e., each record in the resulting NDB is itself some negative database (see Fig. 7). It is important to point out that all NDB_{A_i} 's are the same size (and are thus indistinguishable by this measure) and that some may represent the empty (positive) set.

Compare this scheme to the method described in [19,17] and the examples in Fig. 1, where a monolithic NDB represents all of DB . First, there is additional information leakage⁵, as the size of the underlying DB can be bounded by the number of records (NDB_{A_i} 's) in NDB —a bound, since NDB may contain any number of records that represent the empty set. Second, a NDB created in this manner is much easier to update: inserting a string A_i into DB is implemented as finding which records in NDB represent A_i and removing them; deleting A_i

⁵ Determining the size of DB from a hard-to-reverse NDB is an intractable problem.

DB	NDB_0	NDB_4	NDB_5	NDB_6
000	*1*	*1*	0**	0**
100	1**	**1	**0	*1*
101	0*1	000	*11	10*

Fig. 7. A sample DB with possible NDB_{A_i} (NDB_0 represents the empty set). The final NDB collects all NDB_{A_i} 's. Note that the output of the algorithm presented in Sect. 3 generates NDB_{A_i} 's with exactly three specified bits per record and does not exactly represent $U-DB$; the present example, however, serves to illustrate the non-monolithic structure of the final NDB .

from DB amounts to generating its corresponding NDB_{A_i} and appending it as a record to NDB . The result is a database in which updates take linear time (or better as discussed below) and whose size remains linear in $|DB|$. Moreover, our scheme allows many operations to be parallelized, given that the database can be safely divided into subsets of records and the results easily integrated. This contrasts with the databases and update operations presented in [17], where a single “insert into DB ” requires access to all of NDB , runs in $O(l^4|NDB|^2)$ time, and may cause the database to grow exponentially when repeatedly applied. Finally, the nature of updates remain ambiguous to an observer, given that a record can represent the empty set and that different records (different NDB_{A_i} 's) can stand in for the same DB entry.

We foresee other differences between the two schemes as more complex operations such as joins, projections, etc., are investigated in the context of negative representations of data.

4 Related Work

Reference [19] introduced the concept of negative information, presented negative databases ($NDBs$) as a means to compactly represent negative information, and pointed to the potential of $NDBs$ to conceal data. Additional properties of representing information in this way are outlined in [16]. To date, there are three basic algorithms for creating $NDBs$: the Prefix algorithm [19] is deterministic and always creates a NDB that is easy to reverse; the Randomized algorithm [19] is non-deterministic and can theoretically produce hard-to-reverse $NDBs$, but the required settings are unknown; and finally the On-line algorithms [17,18] designed to update $NDBs$ (insert and delete strings) rely on having an already hard-to-reverse NDB for their security.

There are many other topics that relate to the ideas discussed in this paper. Most relevant are the techniques for protecting the contents of databases—database encryption, zero-knowledge sets, privacy-preserving data mining and query restriction—security systems based on \mathcal{NP} -hard assumptions, and one-way functions.

Some approaches for protecting the contents of a database involve the use of cryptographic methods [23,22,8,41], for example, by encrypting each record

with its own key. Zero-knowledge sets [34,38] provide a primitive for constructing databases that have many of the same properties as negative databases; namely, the restriction of queries to simple membership. However, they are based on widely believed cryptographically secure methods (to which *NDBs* are an alternative), require a controlling entity for answering queries, and are difficult to update.

In privacy-preserving data mining, the goal is to protect the confidentiality of individual records while supporting certain data-mining operations, for example, by computing aggregate statistical properties [6,5,4,13,15,41,40]. In one example of this approach (Ref. [6]), relevant statistical distributions are preserved, but the details of individual records are obscured. Negative databases contrast with this, in that they support simple membership queries efficiently, but higher-level queries may be expensive.

Negative databases are also related to query restriction [32,11,13,14,40], where the query language is designed to support only the desired classes of queries. Although query restriction controls access to the data by outside users, it cannot protect from an insider with full privileges inspecting individual records.

Cryptosystems reliant on \mathcal{NP} -complete problems [21] have been previously studied, e.g., the Merkle-Hellman cryptosystem [33], which is based on the general knapsack problem. These systems rely on a series of tricks to conceal the existence of a “trapdoor” that permits retrieving the hidden information efficiently (*NDBs* have no trapdoors); however, almost all knapsack cryptosystems have been broken [37]. There is a large body of work regarding the issues and techniques involved in generating hard-to-solve \mathcal{NP} -complete problems [29,28,37,33] and in particular of SAT instances [35,12]. Much of this work is focused on the case where formulas are generated without knowledge of their specific solutions. Efforts concerned with the generation of hard instances possessing some specific solution, or solutions with some specific property include [30,24,2].

One-way functions [26,36] and one-way accumulators [7,10] take a string or set of strings and produce a digest from which it’s difficult to obtain the original input. One distinction between these methods and negative databases is that the output of a one-way function is usually compact, and the message it encodes typically has a unique representation (making it easy to verify if a string corresponds to a certain digest). Probabilistic encryption studies how a message can be encrypted in several different ways [27,9].

As the availability of data, the means to access it, and its uses increase, so do our requirements for its security and our privacy. There is no single solution for all of our demands, as evidenced by the many methods reviewed in this section; hard-to-reverse *NDBs*, with their unique characteristics, are an addition to this toolbox.

5 Discussion and Conclusions

In this paper we took the work presented in [19,17,16] and addressed some of its practical concerns. In particular, the previous work outlines algorithms that

are expected to generate hard-to-reverse *NDBs* once their parameters are appropriately set; however, no hints on what their values should be or evidence of them generating any hard instances is provided. The present paper introduced a novel and efficient way to generate negative databases that are extremely hard to reverse. The scheme takes advantage of the relationship the negative data representation has with SAT formulae and borrows from that field a technique for generating the database and the means to test its reversal difficulty. The method we adopted creates an inexact negative image of *DB*, in that the resulting *NDB* negatively represents *DB* along with a few additional strings. We addressed this issue with the inclusion of error detecting codes that help distinguish between *DB* and the extra, superfluous strings.

In addition, our design departs significantly from the previous work's construction of negative databases by securing the contents of the database on a per record basis, i.e., we create a hard-to-reverse NDB_{A_i} for each entry A_i in *DB*, the collection of which constitutes our *NDB*. The present work sketched this setup and outlined some of its characteristics; our current efforts include exploring these database constructions and its applications in more detail.

We have also shown how knowledge from the well established field of SAT can be successfully adapted for the creation and evaluation of negative databases, albeit not always straightforwardly—witness our need to introduce error detecting codes. We expect that more tools and techniques will be transferred in the future, and that better technologies for SAT, e.g., harder formulas to solve, will lead to improved techniques for negative databases and vice versa.

Finally, we are optimistic that some of the problems presented by sensitive data can be addressed by tailoring a negative representation to its particular requirements.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation (grants CCR-0331580 and CCR-0311686, and DBI-0309147), Motorola University Research Program, and the Santa Fe Institute.

References

1. D. Achlioptas, Beame, and Molloy. A sharp threshold in proof complexity. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2001.
2. D. Achlioptas, C. Gomes, H. Kautz, and B. Selman. Generating satisfiable problem instances. In *Proceedings of AAAI-00 and IAAI-00*, pages 256–261, Menlo Park, CA, July 30– 3 2000. AAAI Press.
3. D. Achlioptas and Peres. The threshold for random k -SAT is $2^k \log 2 - O(k)$. *JAMS: Journal of the American Mathematical Society*, 17, 2004.
4. N. R. Adam and J. C. Wortman. Security-control methods for statistical databases. *ACM Computing Surveys*, 21(4):515–556, December 1989.
5. D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Symposium on Principles of Database Systems*, pages 247–255, 2001.

6. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
7. J. Cohen Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology—EUROCRYPT '93*, pages 274–285, 1994.
8. G. R. Blakley and C. Meadows. A database encryption scheme which allows the computation of statistics using encrypted data. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 116–122. IEEE CS Press, 1985.
9. M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In George Robert Blakely and David Chaum, editors, *Advances in Cryptology: proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–302, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1985. Springer-Verlag.
10. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology – CRYPTO ' 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2002.
11. F. Chin. Security problems on inference control for sum, max, and min queries. *J. ACM*, 33(3):451–464, 1986.
12. S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In Du, Gu, and Pardalos, editors, *Satisfiability Problem: Theory and Applications*, volume 35 of *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–17. American Mathematical Society, 1997.
13. D. Denning. *Cryptography and Data Security*. AddisonWesley, Reading, MA, 1982.
14. D.E. Denning and J. Schlorer. Inference controls for statistical databases. *Computer*, 16(7):69–82, July 1983.
15. D. Dobkin, A. Jones, and R. Lipton. Secure databases: Protection against user influence. *ACM Transactions on Database Systems*, 4(1):97–106, March 1979.
16. F. Esponda. *Negative Representations of Information*. PhD thesis, University of New Mexico, 2005.
17. F. Esponda, E. S. Ackley, S. Forrest, and P. Helman. On-line negative databases. In *Proceedings of ICARIS*, 2004.
18. F. Esponda, E. S. Ackley, S. Forrest, and P. Helman. On-line negative databases (with experimental results). *International Journal of Unconventional Computing*, 1(3):201–220, 2005.
19. F. Esponda, S. Forrest, and P. Helman. Enhancing privacy through negative representations of data. *Technical report, University of New Mexico*, 2004.
20. F. Esponda, S. Forrest, and P. Helman. Negative representations of information. *Submitted to International Journal of Information Security*, 2004.
21. S. Even and Y. Yacobi. Cryptography and np-completeness. In *Proc. 7th Colloq. Automata, Languages, and Programming (Lecture Notes in Computer Science)*, volume 85, pages 195–207. Springer-Verlag, 1980.
22. J. Feigenbaum, E. Grosse, and J. A. Reeds. Cryptographic protection of membership lists. 9(1):16–20, 1992.
23. J. Feigenbaum, M. Y. Liberman, and R. N. Wright. Cryptographic protection of databases and software. In *Distributed Computing and Cryptography*, pages 161–172. American Mathematical Society, 1991.

24. C. Fiorini, E. Martinelli, and F. Massacci. How to fake an RSA signature by encoding modular root finding as a SAT problem. *Discrete Appl. Math.*, 130(2):101–127, 2003.
25. I. P. Gent and T. Walsh. The SAT phase transition. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI'94)*, pages 105–109, 1994.
26. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2000.
27. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
28. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 12–24. ACM Press, 1989.
29. R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. In IEEE, editor, *30th annual Symposium on Foundations of Computer Science, October 30–November 1, 1989, Research Triangle Park, NC*, pages 236–241, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1989. IEEE Computer Society Press.
30. H. Jia, C. Moore, and D. Strain. Generating hard satisfiable formulas by hiding solutions deceptively. In *AAAI*, 2005.
31. H. A. Kautz, Y. Ruan, D. Achlioptas, C. Gomes, B. Selman, and M. E. Stickel. Balance and filtering in structured satisfiable problems. In *IJCAI*, pages 351–358, 2001.
32. N. S. Matloff. Inference control via query restriction vs. data modification: a perspective. In *on Database Security: Status and Prospects*, pages 159–166. North-Holland Publishing Co., 1988.
33. R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IT-24:525–530*, 1978.
34. S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *Proc. FOCS 2003.*, page 80, 2003.
35. D. Mitchell, B. Selman, and H. Levesque. Problem solving: Hardness and easiness - hard and easy distributions of SAT problems. In *Proceeding of (AAAI-92)*, pages 459–465. AAAI Press, Menlo Park, California, USA, 1992.
36. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing: Seattle, Washington, May 15–17, 1989*, pages 33–43, New York, NY 10036, USA, 1989. ACM Press.
37. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In Carl Pomerance and S. Goldwasser, editors, *Cryptology and Computational Number Theory*, volume 42 of *Proceedings of symposia in applied mathematics. AMS short course lecture notes*, pages 75–88. pub-AMS, 1990.
38. R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs for generalized queries on a committed database. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, pages 1041–1053, 2004.
39. P. Shaw, K. Stergiou, and T. Walsh. Arc consistency and quasigroup completion. In *In Proceedings of ECAI98 Workshop on Non-binary Constraints*, 1998.
40. P. Tendick and N. Matloff. A modified random perturbation method for database security. *ACM Trans. Database Syst.*, 19(1):47–63, 1994.
41. P. Wayner. *Translucent Databases*. Flyzone Press, 2002.

Related-Key Rectangle Attack on 42-Round SHACAL-2

Jiqiang Lu^{1,*}, Jongsung Kim^{2,3,**}, Nathan Keller^{4,***}, and Orr Dunkelman^{5,†}

¹ Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

`Jiqiang.Lu@rhul.ac.uk`

² ESAT/SCD-COSIC, Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium

`Kim.Jongsung@esat.kuleuven.be`

³ Center for Information Security Technologies(CIST), Korea University
Anam Dong, Sungbuk Gu, Seoul, Korea

`joshep@cist.korea.ac.kr`

⁴Einstein Institute of Mathematics, Hebrew University
Jerusalem 91904, Israel

`nkeller@math.huji.ac.il`

⁵Computer Science Department, Technion
Haifa 32000, Israel

`orrd@cs.technion.ac.il`

Abstract. Based on the compression function of the hash function standard SHA-256, SHACAL-2 is a 64-round block cipher with a 256-bit block size and a variable length key of up to 512 bits. In this paper, we present a related-key rectangle attack on 42-round SHACAL-2, which requires $2^{243.38}$ related-key chosen plaintexts and has a running time of $2^{488.37}$. This is the best currently known attack on SHACAL-2.

Keywords: Block cipher, SHACAL-2, Differential cryptanalysis, Related-key rectangle attack.

1 Introduction

In 2000, Handschuh and Naccache [7] proposed a 160-bit block cipher SHACAL based on the standardized hash function SHA-1 [19]. In 2001, they then proposed

* This author as well as his work was supported by a Royal Holloway Scholarship and the European Commission under contract IST-2002-507932 (ECRYPT).

** This author was financed by a Ph.D grant of the Katholieke Universiteit Leuven and by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF-2005-213-D00077) and supported by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST2002507932 ECRYPT.

*** This author was supported by the Adams fellowship.

† This author was partially supported by the Israel MOD Research and Technology Unit.

two versions, known as SHACAL-1 and SHACAL-2 [8], where SHACAL-1 is the same as the original SHACAL, while SHACAL-2 is a 256-bit block cipher based on the compression function of SHA-256 [20]. Both SHACAL-1 and SHACAL-2 were submitted to the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) project [18] and selected for the second phase of the evaluation; however, in 2003, SHACAL-1 was not recommended for a NESSIE portfolio because of concerns about its key schedule, while SHACAL-2 was selected to be in the NESSIE portfolio.

The published cryptanalytic results on SHACAL-2 are as follows: Hong *et al.* presented an impossible differential attack [2] on 30-round SHACAL-2 [9] and Shin *et al.* presented a differential-nonlinear attack on 32-round SHACAL-2 [21], which is a variant of the differential-linear attack [15]. Shin *et al.* also presented a square-nonlinear attack on 28-round SHACAL-2. Recently, Kim *et al.* [14] presented a related-key differential-nonlinear attack on 35-round SHACAL-2 and a related-key rectangle attack on 37-round SHACAL-2, where the latter attack is based on a 33-round related-key rectangle distinguisher. As far as the number of the attacked rounds is concerned, the Kim *et al.*'s related-key rectangle attack on 37-round SHACAL-2 is the best cryptanalytic result on SHACAL-2, prior to the work described in this paper.

Like the amplified boomerang attack [11] and the rectangle attack [3,4], the related-key rectangle attack [5,10,13] is also a variant of the boomerang attack [22]. As a result, it shares the same basic idea of using two short differentials with larger probabilities instead of a long differential with a smaller probability, but requires an additional assumption that the attacker knows the specific differences between one or two pairs of unknown keys. This additional assumption makes it very difficult or even infeasible to conduct in many cryptographic applications, but as demonstrated in [12], some of the current real-world applications may allow for practical related-key attacks [1], say key-exchange protocols and hash functions.

In this paper, based on relatively low difference propagations for the first several rounds in the key schedule of SHACAL-2, we explore a 34-round related-key rectangle distinguisher. We also introduce a differential property in SHACAL-2 such that we can apply the exploited “early abort” technique to discard some disqualified candidate quartets earlier than usual. Relying on the 34-round distinguisher and the “early abort” technique, we mount a related-key rectangle attack on 40-round SHACAL-2 when used with a 512-bit key. Finally, based on several more delicate observations, we eventually mount a related-key rectangle attack on 42-round SHACAL-2, which requires $2^{243.38}$ related-key chosen plaintexts and has a running time of $2^{488.37}$.

The rest of this paper is organized as follows: In the next section, we briefly describe some notation, the SHACAL-2 cipher and the related-key rectangle attack. In Sect. 3, we introduce four properties in SHACAL-2. In Sect. 4, we present our related-key rectangle attacks on 40 and 42-round SHACAL-2, respectively. Sect. 5 concludes this paper.

2 Preliminaries

2.1 Notation

The following notation will be used throughout this paper:

- \oplus : the bitwise logical exclusive OR (XOR) operation
- $\&$: the bitwise logical AND operation
- \boxplus : the addition modulo 2^{32} operation
- \neg : the complement operation
- e_j : a 32-bit word with zeros in all positions but bit j ($0 \leq j \leq 31$)
- $e_{i_1, \dots, i_j} : e_{i_1} \oplus \dots \oplus e_{i_j}$
- $e_{j, \sim}$: a 32-bit word that has 0's in bits 0 to $j - 1$, 1 in bit j and unconcerned values in bits $(j + 1)$ to 31

2.2 The SHACAL-2 Cipher

SHACAL-2 [8] uses the compression function of SHA-256 [20], where the plaintext enters the compression function as the chaining value, and the key enters the compression function as the message block. Its encryption procedure can be described as follows:

1. The 256-bit plaintext P is divided into eight 32-bit words $A^0, B^0, C^0, D^0, E^0, F^0, G^0$ and H^0 .
2. For $i = 0$ to 63:

$$T_1^{i+1} = K^i \boxplus \Sigma_1(E^i) \boxplus Ch(E^i, F^i, G^i) \boxplus H^i \boxplus W^i,$$

$$T_2^{i+1} = \Sigma_0(A^i) \boxplus Maj(A^i, B^i, C^i),$$

$$H^{i+1} = G^i,$$

$$G^{i+1} = F^i,$$

$$F^{i+1} = E^i,$$

$$E^{i+1} = D^i \boxplus T_1^{i+1},$$

$$D^{i+1} = C^i,$$

$$C^{i+1} = B^i,$$

$$B^{i+1} = A^i,$$

$$A^{i+1} = T_1^{i+1} \boxplus T_2^{i+1}.$$
3. The ciphertext is $(A^{64}, B^{64}, C^{64}, D^{64}, E^{64}, F^{64}, G^{64}, H^{64})$,

where K^i is the i -th round key, W^i is the i -th round constant¹, and the four functions $Ch(X, Y, Z)$, $Maj(X, Y, Z)$, $\Sigma_0(X)$ and $\Sigma_1(X)$ are defined as follows, respectively,

$$Ch(X, Y, Z) = (X \& Y) \oplus (\neg X \& Z),$$

$$Maj(X, Y, Z) = (X \& Y) \oplus (X \& Z) \oplus (Y \& Z),$$

$$\Sigma_0(X) = S_2(X) \oplus S_{13}(X) \oplus S_{22}(X),$$

$$\Sigma_1(X) = S_6(X) \oplus S_{11}(X) \oplus S_{25}(X),$$

where $S_j(X)$ represents right rotation of X by j bits.

¹ In the specifications of [8,20] the term K^i is used for the round constant, and the term W^i is used for the round subkey. In this paper, we use the more standard notation.

The key schedule of SHACAL-2 takes as input a variable length key of up to 512 bits. Shorter keys can be used by padding them with zeros to produce a 512-bit key string; however, the proposers recommend that the key should not be shorter than 128 bits. The 512-bit user key K is divided into sixteen 32-bit words K^0, K^1, \dots, K^{15} , which are the round keys for the initial 16 rounds. Finally, the i -th round key ($16 \leq i \leq 63$) is generated as

$$\begin{aligned} K^i &= \sigma_1(K^{i-2}) \boxplus K^{i-7} \boxplus \sigma_0(K^{i-15}) \boxplus K^{i-16}, \\ &\text{with } \sigma_0(X) = S_7(X) \oplus S_{18}(X) \oplus R_3(X), \\ &\quad \sigma_1(X) = S_{17}(X) \oplus S_{19}(X) \oplus R_{10}(X), \end{aligned} \quad (1)$$

where $R_j(X)$ represents right shift of X by j bits².

2.3 The Related-Key Rectangle Attack

The related-key rectangle attack [5,10,13] treats the block cipher $E : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$ as a cascade of two sub-ciphers $E = E^1 \circ E^0$. It assumes that there exists a related-key differential $\alpha \rightarrow \beta$ with probability p_β^* for E^0 (i.e., $Pr[E_K^0(X) \oplus E_{K^*}^0(X^*) = \beta | X \oplus X^* = \alpha] = p_\beta^*$), where K and K^* are two related keys with a known difference, and a regular differential $\gamma \rightarrow \delta$ with probability q_γ for E^1 (i.e., $Pr[E_K^1(X) \oplus E_K^1(X^*) = \delta | X \oplus X^* = \gamma] = Pr[E_{K^*}^1(X) \oplus E_{K^*}^1(X^*) = \delta | X \oplus X^* = \gamma] = q_\gamma$). In our attack on SHACAL-2 we use a related-key differential for the first sub-cipher and a regular differential for the second sub-cipher, i.e., our second differential has no key difference. Note that the related-key rectangle attack can also use related-key differentials for both the sub-ciphers in similar ways.

Let a quartet of plaintexts be denoted by (P_i, P_i^*, P_j, P_j^*) with $P_i \oplus P_i^* = P_j \oplus P_j^* = \alpha$, where P_i and P_j are encrypted under E_K , and P_i^* and P_j^* are encrypted under E_{K^*} . Out of N pairs of plaintexts with related-key difference α about $N \cdot p_\beta^*$ pairs have a related-key output difference β after E^0 . These pairs can be combined into about $\frac{(N \cdot p_\beta^*)^2}{2}$ candidate quartets such that each quartet satisfies $E_K^0(P_i) \oplus E_{K^*}^0(P_i^*) = \beta$ and $E_K^0(P_j) \oplus E_{K^*}^0(P_j^*) = \beta$. Assuming that the intermediate values after E^0 distribute uniformly over all possible values, the event $E_K^0(P_i) \oplus E_K^0(P_j) = \gamma$ holds with probability 2^{-n} . Once this occurs, $E_{K^*}^0(P_i^*) \oplus E_{K^*}^0(P_j^*) = \gamma$ holds as well, for $E_{K^*}^0(P_i^*) \oplus E_{K^*}^0(P_j^*) = (E_K^0(P_i) \oplus E_{K^*}^0(P_i^*)) \oplus (E_K^0(P_j) \oplus E_{K^*}^0(P_j^*)) \oplus (E_K^0(P_i) \oplus E_K^0(P_j)) = \beta \oplus \beta \oplus \gamma = \gamma$. As a result, the expected number of the quartets satisfying both $E_K^1(P_i) \oplus E_K^1(P_j) = \delta$ and $E_{K^*}^1(P_i^*) \oplus E_{K^*}^1(P_j^*) = \delta$ is

$$\sum_{\beta, \gamma} \frac{(N \cdot p_\beta^*)^2}{2} \cdot 2^{-n} \cdot (q_\gamma)^2 = N^2 \cdot 2^{-n-1} \cdot (\hat{p}^* \cdot \hat{q})^2,$$

where $\hat{p}^* = \sqrt{\sum_{\beta'} Pr^2(\alpha \rightarrow \beta')}$ and $\hat{q} = \sqrt{\sum_{\gamma'} Pr^2(\gamma' \rightarrow \delta)}$.

² We alert the reader to the somewhat confusing notation of $S(\cdot)$ as cyclic rotation and of $R(\cdot)$ as a shift operation.

On the other hand, for a random cipher, the expected number of right quartets is about $\frac{N^2}{2} \cdot 2^{-2n} = N^2 \cdot 2^{-2n-1}$. Therefore, if $\widehat{p} \cdot \widehat{q} > 2^{-n/2}$ and N is sufficiently large, the related-key rectangle distinguisher can distinguish between E and a random cipher.

3 Properties in SHACAL-2

Property 1. (from [21]) Let $Z = X \boxplus Y$ and $Z^* = X^* \boxplus Y^*$ with X, Y, X^*, Y^* being 32-bit words. Then, the following properties hold:

1. If $X \oplus X^* = e_j$ and $Y = Y^*$, then $Z \oplus Z^* = e_{j,j+1,\dots,j+k-1}$ holds with probability $\frac{1}{2^k}$ ($j < 31, k \geq 1$ and $j+k-1 \leq 30$). In addition, in case $j = 31$, $Z \oplus Z^* = e_{31}$ holds with probability 1.
2. If $X \oplus X^* = e_j$ and $Y \oplus Y^* = e_j$, then $Z \oplus Z^* = e_{j+1,\dots,j+k-1}$ holds with probability $\frac{1}{2^k}$ ($j < 31, k \geq 1$ and $j+k-1 \leq 30$). In addition, in case $j = 31$, $Z = Z^*$ holds with probability 1.
3. If $X \oplus X^* = e_{i,\sim}$, $Y \oplus Y^* = e_{j,\sim}$ and $i > j$, then $Z \oplus Z^* = e_{j,\sim}$ holds.

A more general description of this property can be obtained from the following theorem in [16],

Theorem 1. Given three 32-bit differences ΔX , ΔY and ΔZ . If the probability $Pr[(\Delta X, \Delta Y) \stackrel{\boxplus}{\Rightarrow} \Delta Z] > 0$, then

$$Pr[(\Delta X, \Delta Y) \stackrel{\boxplus}{\Rightarrow} \Delta Z] = 2^{-s},$$

where the integer s is given by $s = \#\{i | 0 \leq i \leq 30, \text{not}((\Delta X)_i = (\Delta Y)_i = (\Delta Z)_i)\}$.

Property 2. (from [21]) The two functions Ch and Maj operate in a bit-by-bit manner, therefore, each of them can be regarded as a boolean function from a 3-bit input to a 1-bit output. Table 1 shows the distribution probability of XOR differences through them. The first three rows represent the eight possible differences of the 3-bit inputs x, y, z , and the last two rows indicate the differences in the outputs of the two functions, where a “0” (resp., “1”) means that the difference will always be 0 (resp., 1), and a “0/1” means that the difference will be 0 or 1 with probability $\frac{1}{2}$.

Let’s introduce two other properties in SHACAL-2, as follows.

Property 3. Consider the difference propagation between a pair of data for any four consecutive rounds i to $i+3$. If the difference $(\Delta A^i, \Delta B^i, \dots, \Delta H^i)$ just before the i -th round is known, then we can easily learn that:

1. The differences ΔB^{i+1} , ΔC^{i+1} , ΔD^{i+1} , ΔF^{i+1} , ΔG^{i+1} and ΔH^{i+1} just before the $(i+1)$ -th round can be definitely determined, which are equal to ΔA^i , ΔB^i , ΔC^i , ΔE^i , ΔF^i and ΔG^i , respectively.

Table 1. Differential distribution of the functions Ch and Maj

x	0	0	0	1	0	1	1	1
y	0	0	1	0	1	0	1	1
z	0	1	0	0	1	1	0	1
Ch	0	0/1	0/1	0/1	1	0/1	0/1	0/1
Maj	0	0/1	0/1	0/1	0/1	0/1	0/1	1

2. The differences ΔC^{i+2} , ΔD^{i+2} , ΔG^{i+2} and ΔH^{i+2} just before the $(i+2)$ -th round can be definitely determined, which are equal to ΔB^{i+1} , ΔC^{i+1} , ΔF^{i+1} and ΔG^{i+1} , respectively.
3. The differences ΔD^{i+3} and ΔH^{i+3} just before the $(i+3)$ -th round can be definitely determined, which are equal to ΔC^{i+2} and ΔG^{i+2} , respectively.

Property 4. Let the two related keys K and K^* have the difference e_{31} in both the 0-th and 9-th round keys and have all zero difference in the others of the first 16 round keys, then we can conclude by Eq. (1) that the round keys from 16 until 23 (i.e., $K^{16}, K^{17}, \dots, K^{23}$) have all zero differences, for the following equation holds with probability 1,

$$\begin{aligned}
 K^{*16} &= \sigma_1(K^{*14}) \boxplus K^{*9} \boxplus \sigma_0(K^{*1}) \boxplus K^{*0} \\
 &= \sigma_1(K^{14}) \boxplus (K^9 \oplus e_{31}) \boxplus \sigma_0(K^1) \boxplus (K^0 \oplus e_{31}) \\
 &= \sigma_1(K^{14}) \boxplus K^9 \boxplus \sigma_0(K^1) \boxplus K^0 \\
 &= K^{16}.
 \end{aligned}$$

4 Related-Key Rectangle Attacks on Reduced SHACAL-2

In this section, based on Properties 1, 2 and 4, we explore a 34-round related-key rectangle distinguisher, which can be directly used to mount a related-key rectangle attack on 38-round SHACAL-2. Furthermore, by Property 3, we can partially determine whether a candidate quartet is a valid one earlier than usual; if not, we can discard it immediately, which results in less computations in the left steps and may allow us to proceed by guessing one or more round subkeys, depending on how many candidate quartets are remaining. We call this technique “early abort”. In the case for SHACAL-2, we find that the “early abort” technique can allow us to break two more rounds, that is to say, 40-round SHACAL-2 can be broken faster than an exhaustive key search. Finally, based on several delicate observations, we mount a related-key rectangle attack on 40-round SHACAL-2. The details are as follows.

A 34-Round Related-Key Rectangle Distinguisher. The key schedule of SHACAL-2 has low difference propagations for the first several rounds. Particularly, as exploited in [14], if the two related user keys K and K^* have zero

differences in the first 16 rounds ($0 \sim 15$) except the eighth round key K^8 , one can easily learn from Eq. (1) in the key schedule that the keys from rounds 16 until 22 ($K^{16}, K^{17}, \dots, K^{22}$) have all zero differences. Consequently, Kim *et al.* [14] exploited a 23-round related-key differential characteristic³ $\alpha \rightarrow \beta$ for Rounds $0 \sim 22$ with probability 2^{-33} : $(0, 0, e_{6,9,18,20,25,29}, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}) \rightarrow (0, 0, 0, 0, 0, 0, 0)$. This 23-round related-key differential characteristic requires 22 fixed bits in any pair of plaintexts to increase the differential probability for Round 0.

Then, they exploited a 10-round differential characteristic $\gamma \rightarrow \delta$ for Rounds $23 \sim 32$ with probability 2^{-74} : $(0, e_{9,18,29}, 0, 0, e_{31}, e_{6,9,18,20,25,29}, 0, 0) \rightarrow (e_{11,23}, e_{3,14,15,24,25}, e_{5,27}, e_{9,18,29}, e_{31}, 0, 0, 0)$.

As a result, a 33-round related-key rectangle distinguisher with probability $2^{-470} (= (2^{-33} \cdot 2^{-74})^2 \cdot 2^{-256})$ can be obtained by combining these two differentials. Finally, by counting many possible 10-round differentials $\gamma' \rightarrow \delta$ for Rounds $23 \sim 32$, they obtained a lower bound $2^{-464.32} (= (2^{-33} \cdot 2^{-71.16})^2 \cdot 2^{-256})$ for the probability of this 33-round distinguisher. Based on this 33-round related-key rectangle distinguisher, Kim *et al.* presented a related-key rectangle attack on 37-Round SHACAL-2.

However, we find that the property that the 22-th round key is the furthest round key such that all the round keys from Rounds 16 to 22 have all zero differences is just for the case that the two related user keys K and K^* have non-zero difference in only one of the first 16 round keys. If we study the key schedule more delicately, allowing two, three or more round keys of the first 16 round keys have non-zero differences, we can get that the 23-th round key is the furthest round key such that all the round keys from Rounds 16 to 23 have all zero differences, which requires that K and K^* have the difference e_{31} in both the 0-th and 9-th round keys and have all zero differences in the others of the first 16 round keys. This observation has already been introduced as Property 4 in Sect. 3. Thus, we get one more round with a zero subkey difference than Kim *et al.* Moreover, we observe that these related keys K and K^* produce $K^{24} = L_0 \boxplus L_1$ and $K^{*24} = L_0 \boxplus (L_1 \oplus e_{13,24,28})$, respectively, where $L_0 = \sigma_1(K^{22}) \boxplus K^{17} \boxplus K^8$ and $L_1 = \sigma_0(K^9)$.

Now, we face the problem: could these delicate properties of the key schedule incur a 34-round related-key rectangle distinguisher such that its probability is far greater than 2^{-512} ? Our answer is positive.

Note that e_{31} happens to be the difference in the eighth round key K_8 in the Kim *et al.*'s 23-round related-key differential characteristic. It follows that we can append one more round in the beginning of the Kim *et al.*'s 23-round related-key differential characteristic with the first round key difference e_{31} , which results in a 24-round related-key differential characteristic with probability 2^{-66} :

³ We notice that the probability of the second round of the first differential characteristic presented in [14] is 2^{-13} , and not 2^{-11} as claimed. Hence, the 23-round related-key differential characteristic holds with probability 2^{-33} , not 2^{-31} as claimed in [14]. However, it can be repaired with a little more complexity by the way described below. The corrected probability 2^{-33} is used in our paper.

Table 2. The 24-round related-key differential characteristic for E^0 (Rounds 1 to 24) and the preceding differential for E^b (Round 0), where $M = \{6, 9, 18, 20, 25, 29\}$

Round(i)	ΔA^i	ΔB^i	ΔC^i	ΔD^i	ΔE^i	ΔF^i	ΔG^i	ΔH^i	ΔK^i	Prob.
0	0	e_M	e_{31}	\cdot	$e_{9,13,19}$	$e_{18,29}$	e_{31}	\cdot	e_{31}	\cdot
1	0	0	e_M	e_{31}	0	$e_{9,13,19}$	$e_{18,29}$	e_{31}	0	1
2	e_{31}	0	0	e_M	0	0	$e_{9,13,19}$	$e_{18,29}$	0	2^{-12}
3	0	e_{31}	0	0	$e_{6,20,25}$	0	0	$e_{9,13,19}$	0	2^{-7}
4	0	0	e_{31}	0	0	$e_{6,20,25}$	0	0	0	2^{-4}
5	0	0	0	e_{31}	0	0	$e_{6,20,25}$	0	0	2^{-3}
6	0	0	0	0	e_{31}	0	0	$e_{6,20,25}$	0	2^{-4}
7	0	0	0	0	0	e_{31}	0	0	0	2^{-1}
8	0	0	0	0	0	0	e_{31}	0	0	2^{-1}
9	0	0	0	0	0	0	0	e_{31}	e_{31}	1
10	0	0	0	0	0	0	0	0	0	1
\vdots				\vdots					\vdots	\vdots
23	0	0	0	0	0	0	0	0	0	1
24	0	0	0	0	0	0	0	0	\cdot	2^{-6}
25	$e_{13,24,28}$	0	0	0	$e_{13,24,28}$	0	0	0	\cdot	\cdot

$(0, e_{6,9,18,20,25,29}, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}, e_{2,3,7,8,13,16,20,26,30}) \rightarrow (0, 0, 0, 0, 0, 0, 0, 0)$. Similar to the Kim *et al.*'s attack, we can adopt some delicate improvements to conduct a related-key rectangle attack on 38-round SHACAL-2 based on this 24-round related-key differential and our 10-round differential below. Nevertheless, to make maximally use of Property 3, we will use this appended round for a key recovery in our following attack on 40-round SHACAL-2. Further, let's consider the round key difference $K^{24} \oplus K^{*24}$ in Round 24. Obviously, many difference possibilities are caused due to the addition modulo 2^{32} operations in the key schedule. This round key is then taken the addition modulo 2^{32} operation with the output of Round 23. Due to the zero difference in the output of Round 23, we can count over the possibilities for all the additions together when we compute \hat{p}^* in the following. Here, we can add one more round to the end of the Kim *et al.*'s 23-round related-key differential characteristic to obtain a 24-round (1 ~ 24) related-key differential characteristic $\alpha \rightarrow \beta$ with probability 2^{-38} : $(0, 0, e_{6,9,18,20,25,29}, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}) \rightarrow (e_{13,24,28}, 0, 0, 0, e_{13,24,28}, 0, 0, 0)$. See Table 2 for details. Note that our 24-round related-key differential characteristic described in Table 2 requires the following 12-bit conditions on the two inputs to Round 1, $(A^1, B^1, C^1, D^1, E^1, F^1, G^1, H^1)$ and $(A^{*1}, B^{*1}, C^{*1}, D^{*1}, E^{*1}, F^{*1}, G^{*1}, H^{*1})$ whose difference is α :

$$\begin{aligned}
 a_6^1 &= b_6^1, & a_9^1 &= b_9^1, & a_{18}^1 &= b_{18}^1, & a_{20}^1 &= b_{20}^1, \\
 a_{25}^1 &= b_{25}^1, & a_{29}^1 &= b_{29}^1, & a_{31}^1 &= b_{31}^1, & e_9^1 &= 0, \\
 e_{13}^1 &= 0, & e_{18}^1 &= 1, & e_{19}^1 &= 0, & e_{29}^1 &= 1,
 \end{aligned} \tag{2}$$

where a_i^1 , b_i^1 and e_i^1 are the i -th bits of A^1 , B^1 and E^1 , respectively. If the two input values to Round 1 meet the α difference and Eq. (2), we can remove the

Table 3. The 10-round differential characteristic for E^1 (Rounds 25 to 34), where $M' = \{6, 9, 18, 20, 25, 29, 31\}$

Round(i)	ΔA^i	ΔB^i	ΔC^i	ΔD^i	ΔE^i	ΔF^i	ΔG^i	ΔH^i	Prob.
25	e_{31}	e_{31}	$e_{M'}$	0	0	$e_{9,13,19}$	$e_{18,29,31}$	0	2^{-15}
26	e_{31}	e_{31}	e_{31}	$e_{M'}$	0	0	$e_{9,13,19}$	$e_{18,29,31}$	2^{-12}
27	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	0	$e_{9,13,19}$	2^{-7}
28	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	0	2^{-8}
29	0	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	0	2^{-7}
30	0	0	0	0	e_{31}	e_{31}	e_{31}	$e_{6,20,25}$	2^{-4}
31	0	0	0	0	0	e_{31}	e_{31}	e_{31}	1
32	0	0	0	0	0	0	e_{31}	e_{31}	2^{-1}
33	0	0	0	0	0	0	0	e_{31}	1
34	e_{31}	0	0	0	e_{31}	0	0	0	2^{-11}
35	$e_{6,9,18,20,25,29}$	e_{31}	0	0	$e_{6,20,25}$	e_{31}	0	0	.

differential probabilities incurred by the *Ch* and *Maj* functions in Rounds 1 and 2 (for Round 2, only the condition $a_{31}^1 = b_{31}^1$ is used).

On the other hand, we can use the Kim *et al.*'s 10-round differential characteristic for Rounds 25 to 34 to construct a 34-round related-key rectangle distinguisher. However, we explore a more powerful 10-round differential characteristic $\gamma \rightarrow \delta$ for Rounds 25 ~ 34: $(e_{31}, e_{31}, e_{6,9,18,20,25,29,31}, 0, 0, 0, e_{9,13,19}, e_{18,29,31}, 0) \rightarrow (e_{6,9,18,20,25,29}, e_{31}, 0, 0, e_{6,20,25}, e_{31}, 0, 0)$ ⁴, which holds with probability 2^{-65} . See Table 3.

To compute \hat{p}^* (resp., \hat{q}) (defined in Sect. 2.3), we need to sum the square of the probabilities of all the differentials with the input difference α through E^0 (resp., all the differentials with the output difference δ through E^1), which is computationally infeasible. As a countermeasure, to compute \hat{p}^* , we can count some of such possible differentials that have the same first 23-round differences as the 24-round related-key differential characteristic in Table 2. The 192-bit difference $(\Delta B^{25}, \Delta C^{25}, \Delta D^{25}, \Delta F^{25}, \Delta G^{25}, \Delta H^{25})$ in such a possible output difference of Round 24 can be determined to be all 0's by the corresponding 192-bit difference in the input difference to Round 24, therefore, we only need to count the possible 64-bit output difference $(\Delta A^{25}, \Delta E^{25})$ of Round 24. By counting 42 possible differentials, we can compute a lower bound $2^{-37} (\approx (2^{-38.2} + 6 \cdot 2^{-39.2} + 15 \cdot 2^{-40.2} + 20 \cdot 2^{-41.2})^{\frac{1}{2}})$ for the probability \hat{p}^* of the 24-round differentials $\alpha \rightarrow \beta'$. The upper part of Table 4 gathers some of these differences according to their probabilities. Similarly, we can compute a lower bound $2^{-63.38} (= (2 \cdot 2^{-65.2} + 22 \cdot 2^{-66.2} + 32 \cdot 2^{-67.2})^{\frac{1}{2}})$ for the probability \hat{q} of the 10-round differentials $\gamma' \rightarrow \delta$ by counting 56 out of those that have the same last 9-round differential as the 10-round differential in Table 3: $(e_{31}, e_{31}, e_{6,9,18,20,25,29,31}, \Delta D^{25}, 0, e_{9,13,19}, e_{18,29,31}, \Delta H^{25}) \rightarrow (e_{6,9,18,20,25,29}, e_{31}, 0, 0, e_{6,20,25}, e_{31}, 0, 0)$. The lower part of Table 4 lists some of

⁴ Note that this 10-round differential can be also used to improve the Kim *et al.*'s 33-round related-key rectangle distinguisher.

Table 4. Possible differences in E^0 and E^1 with their respective probability

Prob.	$(\Delta A^{25}, \Delta E^{25})$ in E^0
2^{-38}	$(e_{13,24,28}, e_{13,24,28})$
2^{-39}	$(e_{13,14,24,28}, e_{13,24,28}), (e_{13,24,25,28}, e_{13,24,28}), (e_{13,24,28,29}, e_{13,24,28}),$ $(e_{13,24,28}, e_{13,14,24,28}), (e_{13,24,28}, e_{13,24,25,28}), (e_{13,24,28}, e_{13,24,28,29})$
Prob.	$(\Delta D^{25}, \Delta H^{25})$ in E^1
2^{-65}	$(0, 0), (0, e_{31})$
2^{-66}	$(e_9, e_9), (e_{18}, e_{18}), (e_{29}, e_{29}), (0, e_9), (0, e_{13}), (0, e_{18}), (e_{18}, e_{31}), (e_9, e_{31}),$ $(0, e_{19}), (0, e_{29}), (0, e_{9,31}), (0, e_{13,31}), (0, e_{18,31}), (e_{29}, 0), (e_{18}, 0), (e_9, 0),$ $(0, e_{19,31}), (0, e_{29,31}), (e_9, e_{9,31}), (e_{18}, e_{18,31}), (e_{29}, e_{29,31}), (e_{29}, e_{31})$

these $(\Delta D^{25}, \Delta H^{25})$ according to their probabilities. Therefore, we can obtain a lower bound $2^{-456.76} (= (2^{-37} \cdot 2^{-63.38})^2 \cdot 2^{-256})$ for the probability of our 34-round related-key rectangle distinguisher (Rounds 1 to 34).

4.1 Attacking 40-Round SHACAL-2

We are now ready to explain our related-key rectangle attack on 40-round SHACAL-2. Assume that 40-round SHACAL-2 uses related keys K and K^* whose difference is $(e_{31}, 0, 0, 0, 0, 0, 0, 0, 0, e_{31}, 0, 0, 0, 0, 0, 0)$. First, we use the 34-round related-key rectangle distinguisher to obtain a small portion of subkey candidates in Rounds 0, 35, 36, 37, 38 and 39. Second, we do an exhaustive search for the obtained subkey candidates and the remaining key bits to recover the 512-bit related keys K and K^* . In order to apply the 34-round distinguisher to this attack, we need to collect enough input pairs to Round 1 which meet the α difference and Eq. (2). For this, we use enough pairs of plaintext structures. The details of our attack are as follows:

1. Choose $2^{178.38}$ structures S_i of 2^{64} plaintexts $P_{i,l}$ each, $i = 1, 2, \dots, 2^{178.38}$, $l = 1, 2, \dots, 2^{64}$, where in each structure the 192 bits of words A, B, C, E, F, G are fixed. With a chosen plaintext attack scenario, obtain all their corresponding ciphertexts under the key K , denoted $C_{i,l}$.
2. Compute $2^{178.38}$ structures S_i^* of 2^{64} plaintexts each by XORing the plaintexts in S_i with the 256-bit value $(0, e_{6,9,18,20,25,29}, e_{31}, 0, e_{9,13,19}, e_{18,29}, e_{31}, 0)$. With a chosen plaintext attack scenario, obtain all their corresponding ciphertexts under the key K^* .
3. Guess a 32-bit subkey K^0 in Round 0 and compute $K^{*0} = K^0 \oplus e_{31}$. Encrypt each plaintext $P_{i,l}$ through Round 0 with K^0 to get its intermediate value just after Round 0. We denote the encrypted value by $x_{i,l}$. Check if $x_{i,l}$ meets Eq. (2). If yes, compute $x_{i,l}^* = x_{i,l} \oplus \alpha$ and then decrypt $x_{i,l}^*$ through Round 0 with K^{*0} to get its plaintext, denoted by $P_{i,l}^*$. Find $P_{i,l}^*$ in S_i^* . We denote by $C_{i,l}^*$ the corresponding ciphertext for $P_{i,l}^*$.
4. Guess a 96-bit subkey pair $((K^{37}, K^{38}, K^{39}), (K^{*37}, K^{*38}, K^{*39}))$ in Rounds 37, 38 and 39. For the guessed subkey pair, do the following:

- (a) Decrypt all the ciphertexts $C_{i,l}$ through Rounds 37, 38 and 39 with K^{37} , K^{38} and K^{39} to get their intermediate values just before Round 37. We denote these values by $C_{i,l}^{37}$. Keep them in a table. Decrypt all the ciphertexts $C_{i,l}^*$ through Rounds 37, 38 and 39 with K^{*37} , K^{*38} and K^{*39} to get their intermediate values just before Round 37. We denote these values by $C_{i,l}^{*37}$. Keep them in another table.
 - (b) Check if $C_{i_0,l_0}^{37} \oplus C_{i_1,l_1}^{37}$ and $C_{i_0,l_0}^{*37} \oplus C_{i_1,l_1}^{*37}$ belong to $\delta(2)$, for all $1 \leq i_0 < i_1 \leq 2^{178.38}$, $1 \leq l_0, l_1 \leq 2^{64}$ and all $1 \leq i_0 = i_1 \leq 2^{178.38}$, $1 \leq l_0 < l_1 \leq 2^{64}$, where $\delta(2)$ is the set of all the possible differences caused by the δ difference after 2 rounds. Record $(K^0, K^{37}, K^{38}, K^{39})$ and all the qualified quartets and then go to Step 5.
5. Guess a 32-bit subkey pair (K^{36}, K^{*36}) in Round 36. For the guessed subkey pair, do the following:
 - (a) For each remaining quartet $(C_{i_0,l_0}^{37}, C_{i_1,l_1}^{37}, C_{i_0,l_0}^{*37}, C_{i_1,l_1}^{*37})$, decrypt C_{i_0,l_0}^{37} and C_{i_1,l_1}^{37} through Round 36 with K^{36} to get their intermediate values just before Round 36, and decrypt C_{i_0,l_0}^{*37} and C_{i_1,l_1}^{*37} through Round 36 with K^{*36} to get their intermediate values just before Round 36. We denote the decrypted quartet by $(C_{i_0,l_0}^{36}, C_{i_1,l_1}^{36}, C_{i_0,l_0}^{*36}, C_{i_1,l_1}^{*36})$.
 - (b) Check if $C_{i_0,l_0}^{36} \oplus C_{i_1,l_1}^{36}$ and $C_{i_0,l_0}^{*36} \oplus C_{i_1,l_1}^{*36}$ belong to $\delta(1)$, where $\delta(1)$ is the set of all the possible differences caused by the δ difference after 1 round. Record $(K^0, K^{36}, K^{37}, K^{38}, K^{39})$ and all the qualified quartets and then go to Step 6.
 6. Guess a 32-bit subkey pair (K^{35}, K^{*35}) in Round 35. For the guessed subkey pair, do the following:
 - (a) For each remaining quartet $(C_{i_0,l_0}^{36}, C_{i_1,l_1}^{36}, C_{i_0,l_0}^{*36}, C_{i_1,l_1}^{*36})$, decrypt C_{i_0,l_0}^{36} and C_{i_1,l_1}^{36} through Round 35 with K^{35} to get their intermediate values just before Round 35, and decrypt C_{i_0,l_0}^{*36} and C_{i_1,l_1}^{*36} through Round 35 with K^{*35} to get their intermediate values just before Round 35. We denote the decrypted quartet by $(C_{i_0,l_0}^{35}, C_{i_1,l_1}^{35}, C_{i_0,l_0}^{*35}, C_{i_1,l_1}^{*35})$.
 - (b) Check if $C_{i_0,l_0}^{35} \oplus C_{i_1,l_1}^{35} = C_{i_0,l_0}^{*35} \oplus C_{i_1,l_1}^{*35} = \delta$. If there exist more than 5 quartets passing this δ test, record $(K^0, K^{35}, K^{36}, K^{37}, K^{38}, K^{39})$ and go to Step 7. Otherwise, repeat Step 6 with another guessed key pair (if all the possible key pairs for Round 35 are tested, then repeat Step 5 with another guessed key pair for Round 36; if all the possible key pairs for Round 36 are tested, then repeat Step 4 with another guessed key pair for Rounds 37, 38 and 39; if all the possible key pairs for Rounds 37, 38 and 39 are tested, then repeat Step 3 with another guessed key pair for Round 0).
 7. For a suggested $(K^0, K^{35}, K^{36}, K^{37}, K^{38}, K^{39})$, do an exhaustive search for the remaining 320 key bits using trial encryption. If a 512-bit key is suggested, output it as the master key of the 40-round SHACAL-2. Otherwise, run the above steps with another guess of subkey pair.

This attack requires $2^{243.38}$ related-key chosen plaintexts. The required memory for this attack is dominated by Step 4, which is approximately $2^{243.38} \cdot 32 \approx 2^{247.38}$ memory bytes.

The time complexities of Steps 1 and 2 are $2^{243.38}$ 40-round SHACAL-2 encryptions each. The time complexity of Step 3 is about $(2^{242.38} + 2^{230.38}) \cdot 2^{32} \cdot \frac{1}{40} \approx 2^{269.06}$ 40-round SHACAL-2 encryptions, for Eq. (2) has a 12-bit filtering. Moreover, for each guessed subkey pair, we have about $2^{230.38 \times 2} / 2 = 2^{459.76}$ quartets tested in Step 4. Since the decryptions in Step 4 can be done independent of Step 3, Step 4 requires about $2^{231.38} \cdot 2^{192} \cdot \frac{3}{40} \approx 2^{419.64}$ 40-round SHACAL-2 encryptions and about $2^{231.38} \cdot 2^{192} \cdot 2^{32} = 2^{455.38}$ memory accesses.

From the difference δ , we can definitely determine the differences in words C , D , G , and H of every possible difference in the set $\delta(2)$. Moreover, we observe that there are about 2^{28} possible differences in word B and 2^{17} possible differences in F . Hence, there are about $2^{64+28+17} = 2^{109}$ possible differences in $\delta(2)$. It follows that about $2^{459.76} \cdot 2^{(-256+109) \cdot 2} = 2^{165.76}$ quartets are suggested in Step 4. Since Step 5-(a) runs about 2^{288} times (equivalent to the number of guessed subkey pairs), it requires about $2^{165.76} \cdot 4 \cdot 2^{288} \cdot \frac{1}{40} \approx 2^{450.43}$ 40-round SHACAL-2 encryptions. Similarly, $\delta(1)$ and δ additionally have a 64-bit and a 45-bit filterings, so about $2^{165.76} \cdot 2^{-64 \cdot 2} = 2^{37.76}$ and $2^{37.76} \cdot 2^{-45 \cdot 2} = 2^{-52.24}$ quartets (for each wrong guess of subkey pairs) are expected to be suggested in Steps 5 and 6, respectively, and thus Step 6 requires $2^{37.76} \cdot 4 \cdot 2^{352} \cdot \frac{1}{40} \approx 2^{386.43}$ 40-round SHACAL-2 encryptions. By the Poisson distribution $X \sim Poi(\lambda = 2^{-52.24})$, $Pr_X[X > 5] \approx 2^{-323}$, the expected number of wrong subkey pairs suggested in Step 6 is about $2^{-323} \cdot 2^{352} = 2^{29}$. It follows that the time complexity of Step 7 is about $2^{349} (= 2^{29} \cdot 2^{320})$ 40-round SHACAL-2 encryptions. Therefore, the total time complexity of this attack is about $2^{450.43}$ 40-round SHACAL-2 encryptions.

If the guessed subkey pair is right, then the expected number of the quartets suggested in Step 6 is about $2^{459.76} \cdot 2^{-456.76} = 2^3$, for about $2^{459.76}$ quartets are tested in this attack and the 34-round related-key rectangle distinguisher holds with probability $2^{-456.76}$. Thus, the probability that the number of remaining quartets for the right subkey pair is more than 5 is 0.8 by the Poisson distribution, $X \sim Poi(\lambda = 2^3)$, $Pr_X[X > 5] \approx 0.8$. Hence, this attack works with a success probability of 0.8.

4.2 Attacking 42-Round SHACAL-2

We find that the above attack can be improved to break as far as 42-round SHACAL-2 by guessing the differences between the most significant bits of certain related subkey pairs, instead of guessing the values of these most significant bits. Our improved attack is based on the following observations.

Observation 1: If we know the actual values of (A^i, B^i, \dots, H^i) and $(A^{*i}, B^{*i}, \dots, H^{*i})$, and the additive difference between K^{i-1} and K^{*i-1} , then we know the actual values of $(A^{i-1}, B^{i-1}, \dots, G^{i-1})$ and $(A^{*i-1}, B^{*i-1}, \dots, G^{*i-1})$, and the additive difference between H^{i-1} and H^{*i-1} .

Observation 2: If we know the actual values of $(A^{i-1}, B^{i-1}, \dots, G^{i-1})$ and $(A^{*i-1}, B^{*i-1}, \dots, G^{*i-1})$, and the additive difference between H^{i-1} and H^{*i-1} , then we know the actual values of $(A^{i-5}, B^{i-5}, C^{i-5})$ and $(A^{*i-5}, B^{*i-5}, C^{*i-5})$, and the additive difference between D^{i-5} and D^{*i-5} .

Observation 3: The additive difference between 32-bit words X and Y is the same as their XOR difference if $X \oplus Y = 0$ or $X \oplus Y = e_{31}$.

Based on these observations the above attack algorithm can be improved to an attack on 42-round SHACAL-2. Here, we use the early abort technique one step earlier. Let's briefly describe the attack procedure as follows:

- We perform the above Steps 1, 2 and 3.
- In Step 4, we guess a 64-bit subkey pair $((K^{40}, K^{41}), (K^{*40}, K^{*41}))$ and an additive difference between K^{39} and K^{*39} , and then decrypt all the ciphertexts to obtain the actual values of $(A^{39}, B^{39}, \dots, G^{39})$ and $(A^{*39}, B^{*39}, \dots, G^{*39})$, and the additive difference between H^{39} and H^{*39} (by Observation 1). It allows to know (A^{35}, B^{35}, C^{35}) and $(A^{*35}, B^{*35}, C^{*35})$, and the additive difference between D^{35} and D^{*35} (by Observation 2), so we can discard some wrong quartets by checking if the decrypted quartets satisfy the first half of the δ difference. Since it has a 256-bit filtering for the decrypted quartets, about $2^{459.76} \cdot 2^{-256} = 2^{203.76}$ quartets are suggested. This step requires about $2^{64 \cdot 2 + 32} \cdot 2^{231.38} \cdot \frac{7}{42} = 2^{388.80}$ 42-round SHACAL-2 encryptions and $2^{64 \cdot 2 + 64} \cdot 2^{231.38} = 2^{423.38}$ memory accesses.
- In Step 5, we guess a 64-bit subkey pair of (K^{38}, K^{39}) and (K^{*38}, K^{*39}) (note the additive difference between K^{39} and K^{*39} is fixed in the previous step), and then decrypt all the remaining quartets to obtain their input values of round 38. Since H^{38} is the same as E^{35} , we can discard all the quartets which do not satisfy the $e_{6,20,25}$ XOR difference in H^{38} . It has a 64-bit filtering for the decrypted quartets, so about $2^{203.76} \cdot 2^{-64} = 2^{139.76}$ quartets are suggested. This step requires about $2^{64 \cdot 4 + 32} \cdot 2^{203.76 + 2} \cdot \frac{1}{42} = 2^{488.37}$ 42-round SHACAL-2 encryptions.
- In Step 6, we guess an additive difference between K^{37} and K^{*37} to check if the remaining quartets satisfy the e_{31} difference in H^{37} , which is the same as F^{35} . In Step 7, we guess a 64-bit subkey pair of (K^{36}, K^{37}) and (K^{*36}, K^{*37}) (note the additive difference between K^{37} and K^{*37} is fixed in the previous step) to check if the remaining quartets satisfy zero difference in H^{36} , which is the same as G^{35} . In Step 8, we guess a 64-bit subkey pair of (K^{35}, K^{36}) and (K^{*35}, K^{*36}) (note the additive difference between K^{36} and K^{*36} is fixed in the previous step) to check if the remaining quartets satisfy zero difference in H^{35} . We go to the final step with the guessed subkey pair which has more than 5 remaining quartets. Finally, in Step 9, we do an exhaustive search to find the 512-bit master keys. Each of Steps 6, 7, 8 and 9 takes a dramatically less time complexity than Step 5.

Therefore, the time complexity of the attack is dominated by Step 5, which is about $2^{488.37}$ 42-round SHACAL-2 encryptions. Obviously, the attack is faster than an exhaustive key search.

Table 5. Comparison of our result and previous ones on SHACAL-2 when used with a 512-bit key

<i>Type of Attack</i>	<i>Rounds</i>	<i>Data</i>	<i>Time</i>	<i>Memory</i>	<i>Source</i>
Impossible differential	30	744CP	$2^{495.1}$	$2^{14.5}$	[9]
Differential-nonlinear	32	$2^{43.4}$ CP	$2^{504.2}$	$2^{48.4}$	[21]
Square-nonlinear	28	$463 \cdot 2^{32}$ CP	$2^{494.1}$	$2^{45.9}$	[21]
RK differential-nonlinear	35	$2^{42.32}$ RK-CP	$2^{452.10}$	$2^{47.32}$	[14]
RK Rectangle	37^\dagger	$2^{235.16}$ RK-CP	$2^{486.95}$	$2^{240.16}$	[14]
	40	$2^{243.38}$ RK-CP	$2^{448.43}$	$2^{247.38}$	This paper
	42	$2^{243.38}$ RK-CP	$2^{488.37}$	$2^{247.38}$	This paper

RK: Related-Key, CP: Chosen Plaintexts, Memory unit: Byte, Time unit: Encryption
 \dagger : The indicated attack complexity is a corrected one.

Note: We can reduce the time complexity of our attack on 40-round SHACAL-2 in Section 4.1 to about $2^{448.43}$ 40-round SHACAL-2 encryptions by adopting the following two delicate improvements: First, we only guess the least significant 31 bits of the subkey K^0 in Step 3, due to the fact that the most significant bit in the key difference is fixed. Second, we guess the least significant 31 bits of the subkey pairs (K^{36}, K^{*36}) and the difference between their most significant bits to check the $\delta(1)$ test in Step 5, instead of guessing all the 32-bit values of the subkey pairs. In Step 6, we guess the least significant 31 bits of the subkey pairs (K^{35}, K^{*35}) and the difference between their most significant bits to check the δ test. Since the total time complexity of this attack is dominated by Step 5-(a), it is reduced by a factor of 4.

5 Conclusions

In this paper, we exploit a 34-round related-key rectangle distinguisher after finding a delicate property in the key schedule of SHACAL-2. We also introduce a differential property that can allow us to apply the “early abort” technique to discard some disqualified candidate quartets earlier than usual. Based on them, we mount a related-key rectangle attack on 40-round SHACAL-2. Finally, based on several more delicate observations, we improve it to a related-key rectangle attack on 42-round SHACAL-2. Table 5 compares the results obtained in this paper with the previous ones on SHACAL-2 when used with 512 key bits.

Acknowledgments

The authors are very grateful to Jiqiang Lu’s supervisor Prof. Chris Mitchell for his valuable editorial comments and to the anonymous referees for their helpful advice.

References

1. E. Biham, New types of cryptanalytic attacks using related keys, *Advances in Cryptology — EUROCRYPT'93*, T. Hellesest (ed.), Volume 765 of *Lecture Notes in Computer Science*, pp. 398–409, Springer-Verlag, 1993.
2. E. Biham, A. Biryukov and A. Shamir, Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials, *Advances in Cryptology — EUROCRYPT'99*, J. Stern (ed.), Volume 1592 of *Lecture Notes in Computer Science*, pp. 12–23, Springer-Verlag, 1999.
3. E. Biham, O. Dunkelman and N. Keller, The rectangle attack — rectangling the Serpent, *Advances in Cryptology — EUROCRYPT'01*, B. Pfitzmann (ed.), Volume 2045 of *Lecture Notes in Computer Science*, pp. 340–357, Springer-Verlag, 2001.
4. E. Biham, O. Dunkelman and N. Keller, New results on boomerang and rectangle attacks, *Proceedings of FSE'02*, J. Daemen and V. Rijmen (eds.), Volume 2365 of *Lecture Notes in Computer Science*, pp. 1–16, Springer-Verlag, 2002.
5. E. Biham, O. Dunkelman and N. Keller, Related-key boomerang and rectangle attacks, *Advances in Cryptology — EUROCRYPT'05*, R. Cramer (ed.), Volume 3494 of *Lecture Notes in Computer Science*, pp. 507–525, Springer-Verlag, 2005.
6. E. Biham and A. Shamir, *Differential cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
7. H. Handschuh and D. Naccache, SHACAL, *Proceedings of first open NESSIE workshop*, 2000. Archive available at <https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html>
8. H. Handschuh and D. Naccache, SHACAL, NESSIE, 2001. Archive available at <https://www.cosic.esat.kuleuven.be/nessie/tweaks.html>
9. S. Hong, J. Kim, G. Kim, J. Sung, C. Lee and S. Lee, Impossible differential attack on 30-round SHACAL-2, *Proceedings of INDOCRYPT'03*, T. Johansson and S. Maitra (eds.), Volume 2904 of *Lecture Notes in Computer Science*, pp. 97–106, Springer-Verlag, 2003.
10. S. Hong, J. Kim, S. Lee and B. Preneel, Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192, *Proceedings of FSE'05*, H. Gilbert and H. Handschuh (eds.), Volume 3557 of *Lecture Notes in Computer Science*, pp. 368–383, Springer-Verlag, 2005.
11. J. Kelsey, T. Kohno and B. Schneier, Amplified boomerang attacks against reduced-round MARS and Serpent, *Proceedings of FSE'00*, B. Schneier (ed.), Volume 1978 of *Lecture Notes in Computer Science*, pp. 75–93, Springer-Verlag, 2001
12. J. Kelsey, B. Schneier and D. Wagner, Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES, *Advances in Cryptology — CRYPTO'96*, N. Koblitz (ed.), Volume 1109 of *Lecture Notes in Computer Science*, pp. 237–251, Springer-Verlag, 1996.
13. J. Kim, G. Kim, S. Hong, S. Lee and D. Hong, The related-key rectangle attack — application to SHACAL-1, *Proceedings of ACISP'04*, H. Wang, J. Pieprzyk, and V. Varadharajan (eds.), Volume 3108 of *Lecture Notes in Computer Science*, pp. 123–136, Springer-Verlag, 2004.
14. J. Kim, G. Kim, S. Lee, J. Lim and J. Song, Related-key attacks on reduced rounds of SHACAL-2, *Proceedings of INDOCRYPT'04*, A. Canteaut and K. Viswanathan (eds.), Volume 3348 of *Lecture Notes in Computer Science*, pp. 175–190, Springer-Verlag, 2004.
15. S. K. Langford and M. E. Hellman, *Differential-linear cryptanalysis*, *Advances in Cryptology — CRYPTO'94*, Y. Desmedt (ed.), Volume 839 of *Lecture Notes in Computer Science*, pp. 17–25, Springer-Verlag, 1994.

16. H. Lipmaa and S. Moriai, Efficient algorithms for computing differential properties of addition, Proceedings of FSE'01, M. Matsui (ed.), Volume 2355 of Lecture Notes in Computer Science, pp. 336–350, Springer-Verlag, 2001.
17. M. Matsui, Linear cryptanalysis method for DES cipher, Advances in Cryptology — EUROCRYPT'93, T. Helleseht (ed.), Volume 765 of Lecture Notes in Computer Science, pp. 386–397, Springer-Verlag, 1994.
18. NESSIE, <https://www.cosic.esat.kuleuven.be/nessie/>
19. U.S. Department of Commerce, Secure Hash Standard FIPS 180-1, N.I.S.T., 1995.
20. U.S. Department of Commerce, Secure Hash Standard FIPS 180-2, N.I.S.T., 2002.
21. Y. Shin, J. Kim, G. Kim, S. Hong and S. Lee, Differential-linear type attacks on reduced rounds of SHACAL-2, Proceedings of ACISP'04, H. Wang, J. Pieprzyk, and V. Varadharajan (eds.), Volume 3108 of Lecture Notes in Computer Science, pp. 110–122, Springer-Verlag, 2004.
22. D. Wagner, The boomerang attack, Proceedings of FSE'99, L. Knudsen (ed.), Volume 1636 of Lecture Notes in Computer Science, pp. 156–170, Springer-Verlag, 1999.

On the Collision Resistance of RIPEMD-160*

Florian Mendel, Norbert Pramstaller,
Christian Rechberger, and Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Austria
Florian.Mendel@iaik.tugraz.at

Abstract. In this article, the RIPEMD-160 hash function is studied in detail. To analyze the hash function, we have extended existing approaches and used recent results in cryptanalysis. While RIPEMD and RIPEMD-128 reduced to 3 rounds are vulnerable to the attack, it is not feasible for RIPEMD-160. Furthermore, we present an analytical attack on a round-reduced variant of the RIPEMD-160 hash function. To the best of our knowledge this is the first article that investigates the impact of recent advances in cryptanalysis of hash functions on RIPEMD-160.

Keywords: RIPEMD-160, low-weight codewords, hash function, cryptanalysis, collision attack, differential attack.

1 Introduction

Recent results in cryptanalysis show weaknesses in commonly used hash functions, such as RIPEMD, MD5, Tiger, SHA-0, and SHA-1 [1,2,9,11,12,13,14]. Therefore, the analysis of alternative hash functions, like RIPEMD-160, the SHA-2 family, and Whirlpool is of great interest. Since RIPEMD-160 is part of the ISO/IEC 10118-3:2003 standard on dedicated hash functions, it is used in many applications and is recommended in several other standards as an alternative to SHA-1. Based on the similar design of RIPEMD-160, MD5, SHA-1, and its predecessor RIPEMD, one might doubt the security of RIPEMD-160. Therefore, we investigated the impact of recent attack methods on RIPEMD-160 in detail. We are not aware of any other published analysis with respect to collision attacks of the RIPEMD-160 hash function. In the analysis of the RIPEMD-160 hash function we have extended existing approaches using recent results in cryptanalysis. In the analysis, we show that methods successfully used to attack SHA-1 are not applicable to full RIPEMD-160. Furthermore, we use analytical methods to produce a collision in a RIPEMD-160 variant reduced to 3 rounds. However, no attack has been found for the original RIPEMD-160 hash function. In summary, we can state that RIPEMD-160 is secure against known attack methods. Nevertheless, further analysis is required to get a good view on the security of RIPEMD-160.

* The work in this paper has been supported by the Austrian Science Fund (FWF), project P18138.

Table 1. Notation

Notation	Meaning
$A \oplus B$	logical XOR of two bit-strings A and B
m_i	input message word i (32-bits)
w_i	expanded input message word i (32-bits)
$A \ll n$	bit-rotation of A by n positions to the left
$A \gg n$	bit-rotation of A by n positions to the right
$step$	single execution of the step function
$round$	set of consecutive $steps$, has a size of 16 (1 $round = 16 steps$)

The remainder of this article is structured as follows. A description of the RIPEMD-160 hash function is given in Section 2.1. In Section 2.2, we give an overview of existing attacks on RIPEMD, the predecessor of RIPEMD-160. In Section 2.3, the basic attack strategy we use in our analysis is described. Section 3 presents the results of the analysis following this attack strategy. In Section 4, we describe some methods for improving the results of the analysis. Moreover, we present a theoretical attack on a simplified variant of RIPEMD-160 reduced to 3 rounds using analytical methods in Section 5. We conclude in Section 6.

2 Finding Collisions for RIPEMD-160

In this section, we will give a short description of the RIPEMD-160 hash function. We will present the basic strategy we used for the attack on RIPEMD-160 and we will show why existing attacks on RIPEMD are not applicable to RIPEMD-160. For the remainder of the article we will follow the notation given in Table 1.

2.1 Short Description of RIPEMD-160

The RIPEMD-160 hash function was proposed by Hans Dobbertin, Antoon Bosselaers and Bart Preneel in [8] to replace RIPEMD. It is an iterative hash function that processes 512-bit input message blocks and produces a 160-bit hash value. Like its predecessor RIPEMD, it consists of two parallel streams. In each stream the state variables are updated according to the expanded message word w_i and combined with the initial value IV after the last step, depicted in Figure 1. While RIPEMD consists of two parallel streams of MD4, the two streams are designed differently in the case of RIPEMD-160.

In the following, we briefly describe the RIPEMD-160 hash algorithm. The hash function basically consists of two parts: message expansion and state update transformation. A detailed description is given in [8].

Message Expansion. The message expansion of RIPEMD-160 is a permutation of the message words in each round, where different permutations are used for the left and the right stream.

State Update Transformation. The state update transformation starts from a (fixed) initial value IV of five 32-bit registers and updates them in 5 rounds of

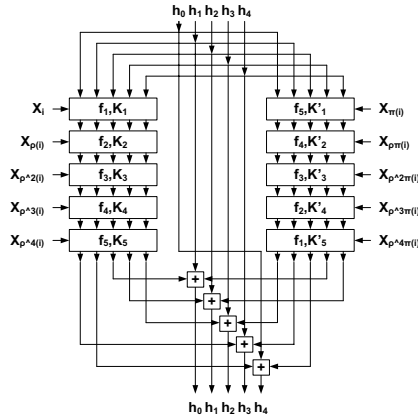


Fig. 1. The RIPEMD-160 compression function

16 steps each by using the expanded message word w_i in step i . Figure 2 shows one step of the state update transformation of RIPEMD-160. The function f is different in each round. f_j is used for the j -th round in the left stream, f_{6-j} is used for the j -th round in the right stream ($j = 1, \dots, 5$). A step constant K_j is added in every step; the constant is different for each round and for each stream. Different rotation values s are used in each step and in both streams. After the last step of the state update transformation, the initial value and the values of the right and the left stream are combined, resulting in the final value of one iteration (feed forward). In detail, the feed forward is a modular addition of the permutations of the IV and the output of the left and right stream (see Figure 1). The result is the final hash value or the initial value for the next message block.

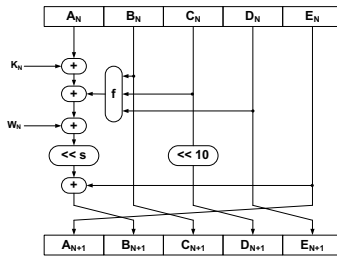


Fig. 2. The step function of RIPEMD-160

For the analysis of RIPEMD-160 in Section 3, we use a linearized variant of the state update transformation. Every addition identified in the hash function is replaced by an XOR and the nonlinear functions f_2, f_3, f_4, f_5 are approximated by a 3-input XOR; f_1 is already an XOR.

2.2 Existing Attacks on the Predecessor RIPEMD

In this section, we will discuss the results in cryptanalysis of RIPEMD, the predecessor of RIPEMD-160. We will describe the attack of Dobbertin and Wang *et al.* and discuss why these attacks are not applicable to RIPEMD-160. A detailed description of both attack strategies is given in [6].

Attack of Dobbertin [7]. In 1997, Hans Dobbertin presented an attack on RIPEMD reduced to 2 rounds with complexity about 2^{31} hash computations. The basic idea of the attack is to find an inner collision for the compression function using a very simple input differential pattern (having only a difference in one message word m_i). Hence, there are differences in the state variables after step i . Since m_i has to be applied in the second round as well, it is chosen in such a way that the differences in state variables cancel out and the remaining steps are equal. Once an inner collision has been found, the remaining free variables have to be determined to meet the *IV* by calculating backward from step i in both streams.

In the attack, Dobbertin uses modular differences to describe the whole hash function by a system of equations. In general, such a system is too large to be solved, but Dobbertin used several constraints to extremely simplify the system such that it becomes solvable in practice. In the attack, he exploits the fact that the left and the right stream of RIPEMD are quite similar. A detailed description of the attack is given in [7].

However, applying the attack to RIPEMD-160 is impractical. Due to the different permutation and rotation values used in the left and the right stream of RIPEMD-160 and due to the increased number of rounds, the system of equations would be too large to be solvable in practice.

Attack of Wang *et al.* [12]. In 2004, Wang *et al.* presented collision attacks on MD4, MD5, and RIPEMD. The attack on RIPEMD has a complexity of about 2^{18} hash computations. The basic idea of all attacks is to use differences in more than one message word to find an inner collision within a few steps in the last round and then find a suitable characteristic for the remaining steps. Hash functions with only 3 rounds seem to be vulnerable to this method in general. Hash functions with more than 3 rounds can only be broken if it is possible to exploit weaknesses of the design [6]. For instance, in the case of RIPEMD, Wang *et al.* take advantage of the similar design of the two streams of the hash function. Since the permutation and rotation values are equal for both streams, it is sufficient to find a collision-producing characteristic for one stream (3 rounds) and apply it simultaneously to both streams. Nevertheless, the number of necessary conditions increases for two streams. Hence, it is more likely to have contradicting conditions. In fact, Wang *et al.* reported that among 30 selected collision-producing characteristics only one can produce the real collision.

However, due to different permutation and rotation values in the left and the right stream of RIPEMD-160 and the increased number of rounds (each stream has 5 rounds), this attack is not applicable to RIPEMD-160.

2.3 Our Attack Strategy

In the following, we will present our attack strategy against RIPEMD-160 based on recent results in cryptanalysis of SHA-1. All attacks basically use the same strategy:

1. Find a collision-producing characteristic that holds with high probability.
2. Find values for the message bits such that the message follows the characteristic.

There are several methods for finding a characteristic, *i.e.* the propagation of input differences through the compression function of the hash function. In the following, we will describe the method of Chabaud and Joux [5] and the method of Wang *et al.* [15] which is used in their attack on SHA-1.

Method of Chabaud and Joux [5]. In 1998, Chabaud and Joux presented an attack on the SHA-0 hash function. In this attack a linearization of the hash function was used to obtain a characteristic (in this paper referred to as *L-characteristic*). The probability that the characteristic holds in the original hash function is related to the Hamming weight of the characteristic. In general, a characteristic with low Hamming weight has a higher probability than one with a high Hamming weight.

Remark 1. *For the first steps, the probability of the characteristic is not important, because the conditions that have to be satisfied such that the characteristic holds can be easily fulfilled for these steps [5].*

Method of Wang *et al.* [15]. Considering the recent results of Wang *et al.*, it seems to be a good approach to use a general (possibly non-linear) characteristic for the first 16 steps and a characteristic that follows the linear approximation for the remaining steps. This is shown in Figure 3. For the remainder of this article the first 16 steps are referred to as V_1 and the remaining steps are referred to as V_2 . The basic idea of this method is to maximize the probability of the *L-characteristic* in V_2 and to ignore the probability of the characteristic in V_1 . This is based on the fact that the probability of V_1 can be neglected (see Remark 1).

Observation 1. *Wang's method to find a characteristic for the hash function can be generalized as follows:*

1. *Find an L-characteristic with good probability resulting in a pseudo-collision for V_2 .*
2. *Find a general characteristic for V_1 to turn a pseudo-collision into a collision.*

Observation 2. *Multi-block messages can be used to turn near-collisions into collisions.*

Since Biham and Chen observed in [1] that near-collisions are easier to find than collisions, we will use Observation 2 in Section 3.3 to improve the attack.

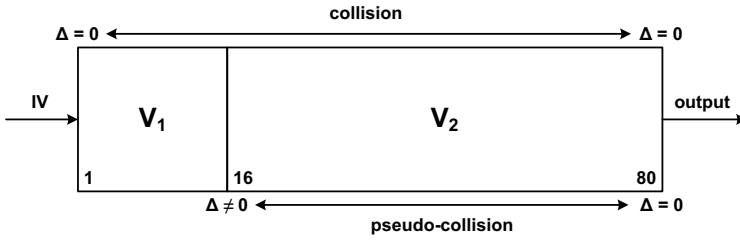


Fig. 3. Attack method of Wang *et al.*

3 Finding an L -characteristic with Good Probability

Finding an L -characteristic for V_2 with good probability is the most important part of the attack. Since the first 16 steps (V_1) can be fulfilled by using message modification techniques [13,14] and neutral bits [1], the attack complexity only depends on the probability of the chosen L -characteristic in V_2 . A common approach to find an L -characteristic with good probability is to search for one with low Hamming weight. In [10,11], algorithms from coding theory were used to obtain an L -characteristic for SHA-1 with low Hamming weight, *i.e.* an L -characteristic with good probability. Even if these algorithms are probabilistic and do not guarantee to find the best L -characteristic, they are expected to produce good results as they did in the case of SHA-1.

For the remainder of this article, we will only give the Hamming weight of the state variable A of the L -characteristic, since this gives us a good heuristic for its probability. More precisely, we use $2^{-2 \cdot \text{HW}(A)}$, where $\text{HW}(A)$ is the Hamming weight of the state variable A . Note that this is a quite conservative method to estimate the probability of the L -characteristic. The probability might be lower in practice.

3.1 Collision and Near-Collision Producing Characteristics

To find a collision-producing characteristic with good probability (low Hamming weight), we use algorithms from coding theory like it is done in [10,11] for SHA-1. To construct the generator matrix G , we use the linearized variant of the state update transformation having zero differences as input in the first step and forcing zero differences after the feed forward (a collision). To keep the generator matrix and the search space small, only state variable A of each step is used. B_i, C_i, D_i, E_i and W_i of step i can be reconstructed from A_i, \dots, A_{i+5} . The Hamming weight of the codewords found and hence the attack complexity is too high for an attack on RIPEMD-160. In Appendix C, the Hamming weight of the codewords found for RIPEMD-160, RIPEMD-128 and round-reduced variants is shown. Considering these results, we conclude that the final attack complexity would be too high for an attack.

Table 2. Hamming weight of A using an NL -characteristic in V_1

	type	Hamming weight	projection*	steps	stream
RIPEMD - 160	pseudo-collision	1471	704	16–80	both
	near-pseudo-collision	907	480	16–80	both
	pseudo-collision	657	352	16–80	left
	pseudo-collision	665	352	16–80	right
	pseudo-collision	959	384	16–64	both
	near-pseudo-collision	675	352	16–64	both
	pseudo-collision	432	192	16–64	left
	pseudo-collision	424	192	16–64	right
	pseudo-collision	458	256	16–48	both
	near-pseudo-collision	428	224	16–48	both
	pseudo-collision	187	96	16–48	left
	pseudo-collision	180	128	16–48	right
RIPEMD - 128	pseudo-collision	659	448	16–64	both
	near-pseudo-collision	561	448	16–64	both
	pseudo-collision	298	256	16–64	left
	pseudo-collision	311	192	16–64	right
	pseudo-collision	178	-	16–48	both
	near-pseudo-collision	18	-	16–48	both
	pseudo-collision	28	-	16–48	left
	pseudo-collision	10	-	16–48	right
RIPEMD	pseudo-collision	20	-	16–48	both

(*)Results achieved by using a projection as described in Section 4.

3.2 Pseudo-collision Producing Characteristics

Since we assume that we are able to turn a pseudo-collision into a collision within V_1 (see Observation 1), we can extend the low-weight search to pseudo-collisions in V_2 . As we want zero differences in the end (after the feed forward) the generator matrix G is constructed by going backwards in V_2 , having zero differences after the feed forward. More precisely, this is done by going backwards in the left and the right stream using the linearized inverse state update transformation. We have a difference δ^L in the left stream and a difference δ^R in the right stream after step 80, where the differences δ^L and δ^R cancel out due to the feed forward.

Table 2 lists the Hamming weight of the codewords found for RIPEMD-160, RIPEMD-128 and round-reduced variants. Note that this weight includes the weight of variable A in the left and the right stream without considering the weight of the first 16 steps. As can be seen in Table 2, we found a codeword for RIPEMD with weight of 20, which might be low enough for an attack following the attack strategy described in this article. Based on the assumed heuristic, we estimate the final attack complexity to be $2^{2 \cdot 20}$. Since the heuristic for the estimation of the attack complexity is quite conservative, the final attack complexity might be higher in practice. Note that the round-reduced variant of the left and the right stream of RIPEMD-128 is very close to an MD4 computation. This explains the low Hamming weight of the codewords found. The results of the left and the right stream differ, because different permutations are used in

the message expansion for both streams. However, the probability of the found *L-characteristic* is too low for an attack on RIPEMD-160 following the strategy described in Section 2.3.

3.3 Near-Pseudo-collision Producing Characteristics

The results of Section 3.2 can be further improved by extending our search to near-collisions. In [13], Wang *et al.* show how this can be done for SHA-1 by using 2 message blocks. They use different characteristics in V_1 , but the same *L-characteristic* in V_2 in both blocks. Due to the permutation of the state variables of the left and the right stream before the addition of both streams and the initial value in the feed forward, we would need 5 instead of 2 message blocks to turn a near-collision into a collision for RIPEMD-160 if we use the same *L-characteristic* in V_2 in each message block.

The results of the low-weight search are shown in Table 2. We found a codeword with weight of 18 for RIPEMD-128 reduced to 3 rounds which is comparable to the result of RIPEMD for a pseudo-collision. However, the Hamming weight of the codewords for RIPEMD-160 is still too high for a reasonable attack complexity. This has several reasons:

- The search space is very large and the problem of finding low-weight codewords in a linear code is NP-hard.
- We do not know any lower bound for the Hamming weight in the code defined by the generator matrix G .
- The search algorithms are probabilistic and certain parameters need to be tuned to optimize the performance. While there exist guidelines, which values to choose for a random code [4], we do not know which values would be optimal in the case of RIPEMD-160.

4 Improving Search Algorithms

Considering the results from the previous section, we have to think about improvements of the probabilistic algorithms. There are several possibilities to increase the speed (success probability for finding a codeword with low Hamming weight) of the algorithms.

4.1 Optimization of the Algorithms/Implementation

Since these algorithms are well known and have been studied by many researchers, we can assume that they are almost optimal in the general case (for a random code). There is still space for some optimizations in the implementation of the algorithms, but the speedup we can obtain is not significant enough.

4.2 Reducing the Search Space

Reducing the search space might be the best way to increase the speed of the probabilistic algorithms we used in the analysis. Since the code describing the

linearized hash function is not a random code, its structure can be exploited to reduce the search space, *i.e.* size of the generator matrix describing the linear code. This method was successfully used for SHA-1. It was observed that differences in the expanded message words and state variables occur in bands of successive ones [11]. For RIPEMD-160, no structure in the low-weight codewords could be found so far. Nevertheless, several methods can be applied to reduce the size of the generator matrix and/or the search space of the algorithms. Some of these methods are:

1. Restricting the analysis to the left (right) stream of the hash function.
2. Looking at round-reduced variants of RIPEMD-160.
3. Using other linearizations for non-linear functions f_2, f_3, f_4 and f_5 .
4. Forcing zero bits (like it is done in [10] for SHA-1). In detail the search space is reduced by setting certain bits (differences) to zero before doing the low-weight search.
5. Reducing the search space by using a projection, $P(w) = \sum_{i=1}^{32} b_i > 0$, where b_i is the i -th bit of the word w . The main idea is to reduce the search space by looking at words instead of bits. In detail, $P(w)$ is 1 if there are differences in the word w and 0 if there are no differences. This reduces the number of columns and rows of the matrix by a factor of 32.

Some of the methods described in this section substantially increase the quality of the results. While the improvements are marginal for reducing the search space by forcing (random) zero bits in the generator matrix or using other linearizations for f_2, f_3, f_4 and f_5 , the other methods worked quite well as shown in Table 2. On the one hand, codewords with lower Hamming weight can be found by reducing the search space but on the other hand the Hamming weight of the codewords found is still too high for an attack on RIPEMD-160 or round-reduced variants. Therefore, we need other (analytical) methods to improve the results.

5 A Variant of RIPEMD-160

In this section, we will describe an approach using analytical methods to find a characteristic with low Hamming weight through the hash function. Since this is very difficult for the original hash function, we concentrate the analysis on a variant of RIPEMD-160, where the rotation of register C is removed, as shown in Figure 4. For this variant, reduced to 3 rounds, we can find a collision using fixed-points.

5.1 Fixed-Points in the RIPEMD-160 Variant

By removing the rotation of register C , it is possible to construct *fixed-points* in one and two steps of the hash function, where a fixed-point is defined as a fixed differential pattern in a single step or two steps of the RIPEMD-160 variant. In Figure 4, a fixed-point for one step of the RIPEMD-160 variant is shown, while

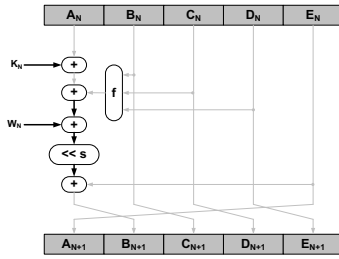


Fig. 4. A fixed-point for one step of the RIPEMD-160 variant

Figure 5 shows fixed-points for two steps of the RIPEMD-160 variant. The gray lines and shadowed rectangles indicate a difference in the MSB. These fixed-points can be used to produce a collision in the RIPEMD-160 variant reduced to 3 rounds with complexity 2^{64} and 2^{51} .

Note that in [3] a similar attack has been applied to MD5 and we can assume that the designers of RIPEMD-160 included the rotation of register C to prevent this kind of attack.

From a Fixed-Point to an Attack. In the analysis, we assume that the conditions for the first 16 steps (V_1) of the hash function can be fulfilled and we can construct differences in the MSB in arbitrary state variables of the left and the right stream after V_1 using a general characteristic. More precisely, if we have differences in the MSB in all state variables of both streams at the first step of V_2 then we can use the fixed-point shown in Figure 4 for the remaining 64 steps in V_2 . The output difference of f with input differences $\delta = (1, 1, 1)$ is 1 or 0, depending on the values of the input variables. Since the difference in the MSB of A_i can be canceled by f , the difference in E_i propagates to B_{i+1} . This results in a collision after the feed forward of the RIPEMD-160 variant. By choosing the differences in the MSB, we reduce the complexity of the attack enormously, since the modular addition behaves linearly for differences in the MSB. So only the conditions for the nonlinear functions f_2, f_3, f_4, f_5 have to be considered for the attack complexity. In detail, one condition has to be fulfilled for the nonlinear functions f_j in each step of the left and the right stream in V_2 .

To cancel a difference in the expanded message word w_i , we exploit the properties of the functions f_j . The output of the functions f_2, f_3, f_4 and f_5 is either 1 or 0 with probability $1/2$ for an input difference $\delta = (1, 1, 1)$, which allows us to cancel differences in the expanded message words in round 2, 3, and 4 of the RIPEMD-160 variant. In the first round of the left stream and in the last round of the right stream, the linear function f_1 is used, making it impossible to cancel a difference there, because f_1 flips with probability 1 for $\delta = (1, 1, 1)$. Since there are differences in all message words in the MSB, f_2, f_3, f_4 have to be blocked in each round of V_2 . We use another (general) characteristic in V_1 . Hence, we have an attack on the RIPEMD-160 variant reduced to 3 rounds. We derive the following set of conditions for round 2 and 3 of the right and the left stream. Note that the conditions are equal for the right and the left stream.

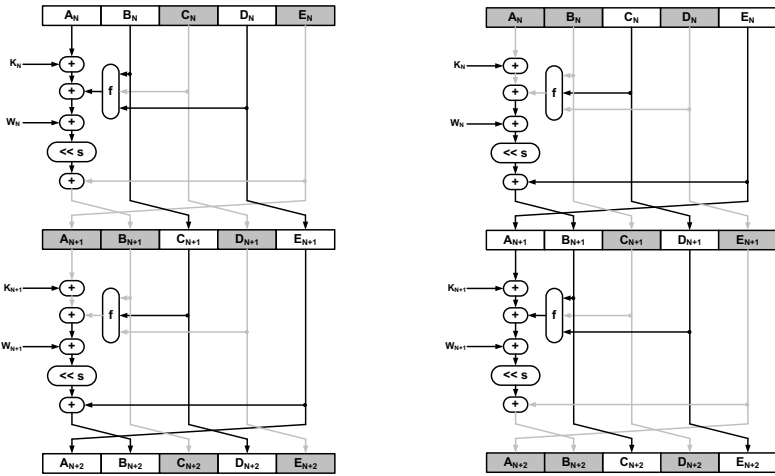


Fig. 5. Two fixed-points for two steps of the RIPEMD-160 variant

$$\begin{aligned}
 B_{i,32} &= \neg C_{i,32} = D_{i,32} & i &= 16 \\
 B_{i,32} &= \neg B_{i-1,32} & i &= 17, \dots, 47
 \end{aligned}$$

This results in a set of 64 conditions (32 for each stream). Satisfying all these conditions with the most naive method (random trials), we get a complexity close to 2^{64} hash computations. Note that no conditions are needed for the modular addition in the feed forward, since we have only differences in the MSB of all state variables of the left and the right stream.

Finding a pseudo-collision in the according RIPEMD-320 reduced to 4 rounds has a complexity of at most 2^{76} hash computations. RIPEMD-320 is an extension of RIPEMD-160 which has the same security level as RIPEMD-160, but produces a hash value of 320 bits. In Appendix B, the conditions for all 4 rounds as well as a sample pseudo-collision on a round-reduced variant (2 rounds) are given.

Improving the Attack Complexity. The attack complexity can be further improved by using one of the fixed-points shown in Figure 5 and by choosing differences in the MSB of w_i , for $i = 1, 4, 6, 7, 10, 11, 12, 15$. Using one of these fixed-points, we can construct an attack on the RIPEMD-160 variant reduced to the first 3 rounds with complexity close to 2^{51} hash computations. By choosing differences in the MSB of w_i , for $i = 1, 4, 6, 7, 10, 11, 12, 15$, only 8 conditions are needed instead of 16 in round 3 of the left and the right stream. This is due to the fact that the differences in the message words are chosen in such a way that only the even or odd words of the left and the right stream have differences in the MSB. Hence, the number of conditions is reduced from 64 to 48. In more detail, if f_3 flips for an input $\delta = (0, 1, 0)$, then it also flips in the next step with input $\delta = (1, 0, 1)$. Hence, round 3 has a probability of 2^{-8} and not 2^{-16} as one may expect. Since we need 5 message blocks to have a collision after the last block, the final attack complexity is $2^{48} \cdot 5$. Since all the differences in the state

variables are in the MSB, no additional conditions have to be fulfilled for the feed forward. Note that the same L -characteristic is used in each message block and only the general characteristic is different for each block. The conditions for the used L -characteristic are given in Appendix A.

5.2 Attack on the RIPEMD-160 Variant Using Fixed-Points

Since we assume that we use a general characteristic in V_1 (first round) to obtain the desired target differences at the input of the first step of V_2 , we have an attack on the RIPEMD-160 variant reduced to the first 3 rounds using one of the fixed-points described before. The attack works as follows:

1. Choose differences in the MSB in message words w_i .
2. Use a general characteristic to construct differences in the MSB in the state variables at the input of the first step in V_2 (to match the desired target difference) and fulfill the conditions for the first 16 steps (V_1) using message modification techniques and neutral bits. Note that if more than one message block is needed to produce a collision then this step has to be repeated for each block.
3. Construct the set of conditions for the L -characteristic in V_2 corresponding to the chosen differences in the message words w_i .
4. Fulfill the conditions for V_2 by random trials. The final attack complexity is related to the number of conditions in V_2 .

Using one message block to construct a collision, the attack has complexity 2^{64} and complexity 2^{51} using 5 message blocks. Even though we cannot extend this attack to the full RIPEMD-160 variant, we conjecture that the rotation of state variable C in the state update transformation enhances the security of RIPEMD-160.

6 Conclusion

In this article, we used recent results in the cryptanalysis of hash functions to analyze the security of RIPEMD-160. We combined methods from coding theory with recent attack techniques which were successfully used in the attack on SHA-1. While RIPEMD and RIPEMD-128 reduced to 3 rounds are vulnerable to this kind of attack, the attack is not suitable for RIPEMD-160.

Furthermore, we analyzed a variant of RIPEMD-160, where the rotation of state variable C was removed. We show that for this variant an attack on 3 rounds is possible using fixed-points. Hence, we conclude that the rotation of state variable C enhances the security level of RIPEMD-160.

We found no attack on the original RIPEMD-160 hash function including all 5 rounds. In summary, we state that RIPEMD-160 is secure against known attacks. Neither the attack of Dobbertin or Wang *et al.* on RIPEMD can be extended to RIPEMD-160, nor recent methods used in the cryptanalysis of SHA-1 are applicable to full RIPEMD-160. Even though this paper gives new insights on the security of RIPEMD-160, further analysis is required to get a good view on its security margin.

References

1. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 36–57. Springer, 2005.
3. Bert den Boer and Antoon Bosselaers. Collisions for the Compression Function of MD5. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *LNCS*, pages 293–304. Springer, 1994.
4. Florent Chabaud. On the Security of Some Cryptosystems Based on Error-correcting Codes. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *LNCS*, pages 131–139. Springer, 1995.
5. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462, pages 56–71. Springer, 1998.
6. Magnus Daum. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr Universität Bochum, 2005. Available at <http://www.cits.rub.de/imperia/md/content/magnus/dissmd4.pdf>.
7. Hans Dobbertin. Ripemd with two-round compress function is not collision-free. *J. Cryptology*, 10(1):51–70, 1997.
8. Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *LNCS*, pages 71–82. Springer, 1996.
9. Krystian Matusiewicz and Josef Pieprzyk. Finding good differential patterns for attacks on SHA-1. *Cryptology ePrint Archive*, Report 2004/364, 2004. <http://eprint.iacr.org/>.
10. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.
11. Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
12. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

13. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
14. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
15. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

A Set of Sufficient Conditions for the L -characteristic of the RIPEMD-160 Variant Reduced to 3 Rounds

In this section, we will give the complete set of sufficient conditions for the attack on the RIPEMD-160 variant reduced to 3 rounds using a fixed-point for 2 steps as described in Section 5. For the analysis, we assume that we can find a general characteristic for round 1 such that we have differences in state variable C of the left stream and the right stream in the input of the first step of round 2. Since there are differences in the message words w_i , for $i = 1, 4, 6, 7, 10, 11, 12, 15$, the number of conditions is reduced as described in Section 5. Hence, we derive the following set of equations for the L -characteristic for round 2 and 3 of the right and the left stream.

– Left Stream:

$$\begin{aligned}
 B_{i,32} &= 0 & i &= 18, 26, 28 \\
 B_{i,32} &= 1 & i &= 16, 20, 22, 24, 30, 32, 34, 36, 38, 40, 42, 44, 46 \\
 B_{i,32} &= B_{i-2,32} & i &= 19, 23, 25, 31 \\
 B_{i,32} &= \neg B_{i-2,32} & i &= 17, 21, 27, 29
 \end{aligned}$$

– Right Stream:

$$\begin{aligned}
 B_{i,32} &= 0 & i &= 14, 26, 28, 32, 34, 36, 38, 40, 42, 44, 46 \\
 B_{i,32} &= 1 & i &= 16, 18, 20, 22, 24 \\
 B_{i,32} &= B_{i-2,32} & i &= 17, 19, 23, 25, 29 \\
 B_{i,32} &= \neg B_{i-2,32} & i &= 21, 27 \\
 B_{i,32} &= B_{i-1,32} \oplus B_{i-2,32} & i &= 31
 \end{aligned}$$

B Set of Sufficient Conditions for a Pseudo-collision in a Round-Reduced RIPEMD-320 Variant

In this section, we will give a set of sufficient conditions for a pseudo-collision in a RIPEMD-320 variant. Note that there are no differences in the message words and the IV has differences in the MSB of all words. This would result in an attack complexity of 2^{128} for a pseudo-collision of RIPEMD-320. Since we assume that we can fulfill the first 16 to 20 steps of the right stream (no conditions have to be fulfilled for the first 16 steps in the left stream), the attack complexity would be 2^{108} .

– Left Stream:

$$\begin{aligned}
 B_{i,32} = C_{i,32} = D_{i,32} = 1 & \quad i = 16 \\
 B_{i,32} = B_{i-1,32} & \quad i = 17, \dots, 79
 \end{aligned}$$

– Right Stream:

$$\begin{aligned}
 B_{i,32} = C_{i,32} = D_{i,32} = 1 & \quad i = 1 \\
 B_{i,32} = B_{i-1,32} & \quad i = 2, \dots, 63
 \end{aligned}$$

Below, a message and the according IV is given for a pseudo-collision in the first 2 rounds of the RIPEMD-320 variant, which has a complexity of 2^{28} hash computations. A pseudo-collision for the first 3 rounds would require about 2^{60} hash computations.

i	M								
0-7	1330C95E	D6E82F5D	1902E1F8	040C42B4	F51D77D2	B8EF7ED0	D075FEE3	1CB083FD	
8-15	37246C9D	72205B19	703A3DCD	E7E5AFFD	FD9D1E57	4C64C76F	4B424959	56B11DB4	

i	IV					
0-4	A99DA4B3	257D7E0C	56D85144	8F93F035	79096694	
5-9	58EEE5C0	AA910BAB	BD91DCA9	8D5BE12A	14C72EFO	

C Hamming Weight of Codewords Found for Using an L -characteristic in V_1 and V_2

In this section, we will give the Hamming weight of the codewords found for using an L -characteristic in V_1 and V_2 as described in Section 3.1. In Table 3, the Hamming weight of the codewords found for RIPEMD-160, RIPEMD-128,

and round-reduced variants are shown. Since we assume that it is possible to turn near-collisions into collisions by using multi-block messages (see Observation 2), we can improve the Hamming weight of the codewords found and hence the probability of the characteristic. For a near-collision, the condition of having zero differences after the feed forward can be ignored. The Hamming weight of the codewords found are also shown in Table 3. Note that we only give the Hamming weight after step 16, since the first 16 steps (V_1) can be fulfilled in advance, and only the probability of V_2 is significant for the attack complexity. We conclude that the final attack complexity would be too high for an attack.

Table 3. Hamming weight of A using an L -characteristic in V_1 and V_2

	type	Hamming weight	projection*	steps	stream
RIPEMD - 160	collision	1760	768	16–80	both
	near-collision	1568	768	16–80	both
	collision	895	448	16–80	left
	collision	848	576	16–80	right
	collision	1184	576	16–64	both
	near-collision	1184	576	16–64	both
	collision	608	320	16–64	left
	collision	644	352	16–64	right
	collision	863	384	16–48	both
	near-collision	768	352	16–48	both
	collision	421	160	16–48	left
	collision	414	128	16–48	right
RIPEMD - 128	collision	1303	640	16–64	both
	near-collision	880	512	16–64	both
	collision	602	256	16–64	left
	collision	576	320	16–64	right
	collision	800	256	16–48	both
	near-collision	640	256	16–48	both
	collision	377	64	16–48	left
	collision	374	128	16–48	right

(*)Results achieved by using a projection as described in Section 4.

Blind Ring Signatures Secure Under the Chosen-Target-CDH Assumption

Javier Herranz^{1,*} and Fabien Laguillaumie²

¹ Centrum voor Wiskunde en Informatica (CWI)
Kruislaan 413, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands

`Javier.Herranz@cw.nl`

² Projet TANC - INRIA Futurs
Laboratoire d'informatique (LIX), École polytechnique
91128 Palaiseau cedex - France

`laguillaumie@lix.polytechnique.fr`

Abstract. Blind signatures are a useful ingredient to design secure sophisticated systems like electronic voting or sensitive applications like e-cash. Multi-users signature schemes, like ring or group signatures, are also a useful tool to provide to such systems some properties like scalability, anonymity, (dynamic) group structure, revocation facilities. . . We propose in this article a simple blind ring signature scheme based on pairings on algebraic curves. We formally prove the security (anonymity, blindness and unforgeability) of our scheme in the random oracle model, under quite standard assumptions.

Keywords: blind ring signatures, e-cash systems, provable security.

1 Introduction

Blind signatures were introduced by Chaum [13]. They allow a person to get a message signed by another party without revealing any information about the message to this other party. Blind signatures have been intensively studied since their birth. A precise security model is provided in Pointcheval and Stern's paper [20]. Possible applications of blind signatures can be found in electronic auctions and electronic voting systems. However, the original motivation for the use of such signatures came from e-cash and untraceable payments. Roughly speaking, an electronic coin corresponds to a certain amount of money and it is blindly signed by a bank (therefore, the bank does not know the true value of the coin). It is then withdrawn from the bank, spent by a user, and deposited by a shop.

To make this system more scalable by supporting many banks (to fit with real life scenarios), and to possibly add some other properties like strong anonymity of the signing banks, non linkability of two different signatures, revocation facilities, etc., Lysyanskaya and Ramzan introduced the concept of blind group

* The work of the first author was carried out during the tenure of an ERCIM fellowship.

signatures [17], which combines the concepts of blind signatures and group signatures. Group signatures allow any member of a group to sign a document in such a way that a verifier can confirm that the signature comes from the group, but he does not know which member of the group actually signed the document. The protocol allows for the identity of the signer to be discovered, in case of disputes, by a designated group authority that has some auxiliary information. Group signatures have been introduced by Chaum and van Heyst [14]. Like blind signatures, lots of schemes arose in the literature and one can mention Ateniese, Camenisch, Joye and Tsudik's scheme [2] and Boneh, Boyen and Shacham's pairing-based scheme [8] among the most promising and efficient protocols. The security model for group signatures has been finally properly defined by Bellare, Micciancio and Warinschi in their paper [3]. Ring signatures, introduced by Rivest, Shamir and Tauman [21], are somehow similar to group signatures, but with some important differences: (1) the group is not fixed, but chosen by the actual signer in an ad-hoc way, just before computing the signature; (2) there is no group authority who can recover the identity of the author of a ring signature. Ideally, anonymity in ring and group signature schemes should be satisfied in an unconditional way: no information about the author of a signature must be obtained, even if one has unlimited computational resources. In this way, a signer can be sure that his identity as the author of a signature is perfectly protected for the rest of his life. We refer the reader to Wang's on line bibliography on digital signature [22] for a full overview these different signature schemes.

As we have said before, the first proposed group blind signature scheme is Lysyanskaya and Ramzan's one [17], based on Camenisch and Stadler's group signature scheme with constant size signatures [11]. Applied to the scenario of distributed electronic banking, a central bank behaves as the group authority and monitors the group members, which are banks issuing e-cash. Nguyen, Mu and Varadharajan [19] also proposed a blind variant of Camenisch and Stadler's scheme.

Obviously, combining blind and ring signatures also brings solutions to these scenarios of e-banking, e-voting or e-auctions. Indeed ring signatures provide more spontaneity and flexibility to the design of such systems. Namely, suppose that a client wants some bank to sign some electronic coin corresponding to a certain amount of money; the client can choose ad-hoc a set (or *ring*) of potential signing banks, depending on some conditions (for example, the use that the client is going to make of the obtained coin). If some bank in the ring accepts to sign this coin, it starts running the interactive signing protocol with the client. The bank can therefore preserve its anonymity inside the ring of banks, if desired; on the contrary, if it wants to publicly show its identity, it can simply run a standard (not ring) blind signature scheme, or to use a blind ring signature scheme where the considered ring has this bank as the only member. Summing up, the ring can be chosen by the client or by the actual signer, because of the interactive nature of the protocols, and this increases the number of real-life applications of this kind of schemes.

Only few blind ring signature schemes have been proposed up to now. Chan, Fung, Liu and Wei [12] proposed the first one in 2005. This scheme is obscure and it is unclear who actually engages the different protocols. Furthermore, the proofs provided in the paper are not very convincing. All these facts make us suspect that this scheme does not satisfy some required properties such as blindness or anonymity. Finally, Wu, Zhang, Susilo and Mu have recently described an efficient static blind ring signature [23], with constant signature size and efficient algorithms. In this scheme, each user knows the factorization of an RSA modulus $n_i = p_i q_i$. Basically, the underlying ring signature consists for the signer, given $y = g^{n_1 \dots n_k} \bmod N$ where N is a public RSA modulus of unknown factorization, and g a generator of $(\mathbb{Z}/N\mathbb{Z}^*)^2$, in proving that he knows p_i and $u = g^{q_i \prod_{j \neq i} n_j}$ such that $y = u^{p_i}$, with p_i in a certain range. An external trusted entity is therefore needed, at least in the setup phase of the system, to generate N . Another drawback of the scheme is that anonymity only holds computationally: an adversary with enough computational resources can factorize all the RSA moduli and automatically obtain the identity of the author of each signature. As discussed above, this is not desirable for some applications; maybe a bank does not want its identity to be revealed in the future as the issuer of some (possibly controversial) e-cash. Furthermore, the unforgeability of this scheme relies on strong (and quite debatable) assumptions like the “extended ROS” one, and is proved in the generic group model (which is stronger than the random oracle model). Even if the authors claim that their scheme supports only static groups, we think that this is not true, and that the client who wants to obtain a blind signature can choose the ring of signers in an ad-hoc way. Apparently, authors of [23] consider only static groups to avoid some attacks against blindness. We think that the blindness property definition only makes sense when the two considered signatures involve the same ring of signers; this is independent of the fact that the scheme can be employed for different rings. See more details on this point in Section 3.1, where we propose a formal and quite natural definition for the blindness property of a blind ring signature scheme.

Our Contributions. In this article, we extend Boneh, Gentry, Lynn and Shacham’s pairing-based ring signatures [9] by adding the feature of blindness. This scheme accepts in essence the pairing-based blindness techniques described by Boldyreva in [7]. We analyze the security of the resulting blind ring signature scheme by providing first a suitable model for the required properties: anonymity, blindness and unforgeability. Then we prove the security of our new scheme in the random oracle model, under quite standard assumptions, without using the generic model or ROS-like assumptions. Our scheme suffers from the drawback that the number of computations and the size of signatures grow linearly with the number of signers in the ring. This problem is recurrent and inherent to ring signatures supporting dynamic rings, because the description of the ring is necessary to verify a signature. This description usually consists in the set of public keys, and so the length of the (blind) ring signature is always linear with respect to the number of users. Techniques based on accumulators allow to obtain constant-size ring signatures, see [15], when the same ring is used for many signatures.

Our scheme is advantageous with respect to the solutions employing group signatures because it is dynamic, in the sense that the group is chosen “ad-hoc” by the client who wants to obtain a blind signature. Furthermore, neither interaction among the set of users nor initialization phase are required: each user generates his own secret/public keys in an independent way. Contrary to Wu *et al.*'s scheme in [23], the anonymity property is obtained in an unconditional way, which means that the identity of the author of a signature is perfectly protected. Finally, it is easily implemented and based on simple operations, due to the spectacular progress of pairing-based tools.

The rest of the paper is organized as follows: in Section 2, we recall the basics about bilinear pairings and give the computational assumptions (of the chosen-target problem family) which underlie our scheme, and especially the chosen-target-inverse-CDH problem that we prove equivalent to the traditional chosen-target CDH, used in [7] to prove the unforgeability of the blind signature scheme. Then we precisely define in Section 3 a blind ring signature scheme and the security properties that such a scheme should satisfy. In Section 4, we present our new scheme, and formally prove its security. The conclusions of the work and some open problems are given in Section 5.

2 Bilinear Pairings and Computational Assumptions

In this section, we recall some basic facts about bilinear maps and introduce the computational assumptions needed to prove the security of our scheme.

Definition 1. *Let \mathbb{G} be an additive group of prime order q , generated by some element P . Let \mathbb{H} be a multiplicative group with the same order q .*

A symmetric admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{H}$ satisfies the following three properties:

- i) it is bilinear;*
- ii) it can be efficiently computed for any possible input pair;*
- iii) it is non-degenerate, which means that $e(P, P) \neq 1$.*

The typical way of obtaining such pairings is by deriving them from the Weil or the Tate pairing on (hyper-)elliptic curves over a finite field (see for instance [1]).

The security of blind signature schemes is based, in general, on the hardness of the *chosen-target* versions of standard computational problems, such as chosen-target RSA problem [4] for the scheme in [13], or the chosen-target CDH problem for the scheme in [7].

The *Chosen-Target-CDH problem* is defined as follows: the solver \mathcal{S} receives as input a pair (P, aP) , where P is a generator of \mathbb{G}_1 with prime order q , and $a \in \mathbb{Z}_q$ is a random value. The solver \mathcal{S} has adaptive access to two oracles:

- **target oracle:** this oracle outputs a random element $Z_i \in \mathbb{G}_1$,
- **helper oracle:** this oracle takes as input an element $W_i \in \mathbb{G}_1$ and outputs the element aW_i .

We say that $\mathcal{S} (q_t, q_h, d)$ -solves the Chosen-Target-CDH problem, for $q_t \geq d > q_h$, if it makes q_t and q_h queries, respectively, to the target and helper oracles, and after that it outputs d pairs $((V_1, j_1), \dots, (V_d, j_d))$ such that:

1. all the elements V_i are different,
2. for all $i \in \{1, 2, \dots, d\}$, the relation $V_i = aZ_{j_i}$ is satisfied, where Z_{j_i} is the element output by the target oracle in the j_i -th query.

To fit our purpose, we define a very similar problem, which in fact is equivalent (see Prop. 1) to the Chosen-Target-CDH problem. This new problem, that we call *Chosen-Target-Inverse-CDH problem*, is defined as follows: the solver \mathcal{S}' receives as input a pair $(P', a'P')$, where P' is a generator of \mathbb{G}_1 with primer order q , and $a' \in \mathbb{Z}_q$ is a random value. The solver \mathcal{S}' has adaptive access to two oracles:

- **target oracle:** this oracle outputs a random element $Z_i \in \mathbb{G}_1$,
- **helper oracle:** this oracle takes as input an element $W_i \in \mathbb{G}_1$ and outputs the element $\frac{1}{a'}W_i$.

We say that $\mathcal{S}' (q_t, q_h, d)$ -solves the Chosen-Target-Inverse-CDH problem, for $q_t \geq d > q_h$, if it makes q_t and q_h queries, respectively, to the target and helper oracles, and after that it outputs d pairs $((V_1, j_1), \dots, (V_d, j_d))$ such that:

1. all the elements V_i are different,
2. for all $i \in \{1, 2, \dots, d\}$, the relation $V_i = \frac{1}{a'}Z_{j_i}$ is satisfied, where Z_{j_i} is the element output by the target oracle in the j_i -th query.

Lemma 1. *The Chosen-Target-CDH problem and the Chosen-Target-Inverse-CDH problem are equivalent.*

Proof. We show only one of the implications, since the other one can be proved in an identical way. Let us assume, for example, that there exists \mathcal{S} which (q_t, q_h, d) -solves the Chosen-Target-CDH problem, and let us construct from it a solver \mathcal{S}' which (q_t, q_h, d) -solves the Chosen-Target-Inverse-CDH problem.

\mathcal{S}' receives as input a pair $(P', a'P')$, has access to its target and helper oracles, and wants to solve the Chosen-Target-Inverse-CDH problem. To do this, it initializes the (q_t, q_h, d) -solver \mathcal{S} with input pair $(P, aP) = (a'P', P')$. Note that this means $a = 1/a'$. To obtain from \mathcal{S} a solution of the Chosen-Target-CDH problem, \mathcal{S}' must simulate the environment of \mathcal{S} , by answering all the queries that \mathcal{S} makes to its oracles:

- **target oracle:** when \mathcal{S} makes a query to this oracle, \mathcal{S}' makes a query to its own target oracle, and sends to \mathcal{S} the obtained random element $Z_i \in \mathbb{G}_1$;
- **helper oracle:** when \mathcal{S} makes a query W_i to this oracle, \mathcal{S}' makes the same query W_i to its own helper oracle. By definition, the helper oracle of \mathcal{S}' returns the element

$$\frac{1}{a'}W_i = aW_i.$$

Therefore, \mathcal{S}' sends to \mathcal{S} this value, which is consistent with the answers of a real helper oracle for \mathcal{S} .

After q_t and q_h queries to the respective oracles, \mathcal{S} finally outputs d pairs $((V_1, j_1), \dots, (V_d, j_d))$ such that:

1. all the elements V_i are different,
2. for all $i \in \{1, 2, \dots, d\}$, the relation $V_i = aZ_{j_i} = \frac{1}{a^7}Z_{j_i}$ is satisfied, where Z_{j_i} is the element output by the target oracle in the j_i -th query.

Note that such a list of pairs is a valid solution for the instance $(P', a'P')$ of the Chosen-Target-Inverse-CDH problem that \mathcal{S}' received. Therefore, \mathcal{S}' has (q_t, q_h, d) -solved the Chosen-Target-Inverse-CDH problem. \square

3 Blind Ring Signature Schemes

Given an integer k , a *blind ring signature scheme* BRS with security parameter k consists of the following four algorithms:

- **generation of public parameters:** BRS.Setup is a probabilistic algorithm which takes as input k and outputs public parameters (which include a description of the signature space, hash functions, etc.);
- **key generation:** BRS.KeyGen is a probabilistic algorithm which takes as input the public parameters and outputs a signing key pair (pk_j, sk_j) for a user U_j . The value pk_j is made public, whereas the value sk_j is secretly stored by user U_j .
- **blind ring signature generation:** BRS.Sign is an interactive 2-party protocol which is initialized by a client \mathcal{C} . This client chooses a message M and a ring $\mathcal{U} = \{U_1, \dots, U_n\}$ of users, and engages an interaction with some of the members U_s of the ring, who can use his secret key sk_j as part of the input. We denote as $\mathcal{I}_{\mathcal{C}}$ the secret inputs that client \mathcal{C} uses, and as \mathcal{T}_{sig} the values that are obtained by the signer, during this interaction.

At the end, the private output $\mathcal{O}_{\mathcal{C}}$ for the client is a valid ring signature Σ for the message M and the ring of users \mathcal{U} .

- **Verification of a blind ring signature:** BRS.Verify is a deterministic algorithm which takes as input a message M , a ring of users $\mathcal{U} = \{U_1, \dots, U_n\}$, their public keys pk_1, \dots, pk_n and bit string Σ . The output is 1 if the signature is valid, and 0 otherwise.

A blind ring signature scheme must satisfy 4 requirements:

1. *Correctness* means that a verifier always accepts as valid a signature that has been properly generated by a honest client and a honest signer in the corresponding ring of users.
2. *Anonymity* means that the client has no information about which member of the ring has actually participated in the interactive blind ring signature generation.
3. *Blindness* intuitively means that the users in the ring obtain no information about the message that they are actually signing for the client.

4. *Unforgeability* means that a client is not able to produce $\ell + 1$ valid and different ring signatures if he has queried for at most ℓ executions of the blind ring signature protocol.

We now recall the formal definition of the two last properties.

3.1 Blindness

Blindness of a blind ring signature scheme is defined by a game played between a challenger and an adversary. This adversary \mathcal{B} models the dishonest behaviour of a ring of users who try to distinguish which message (between two messages chosen by them) is being signed in an interactive execution of the signing protocol with a client. The game is as follows:

1. **Setup:** the adversary \mathcal{B} chooses a universe \mathcal{U}^* of users and a security parameter k . The challenger runs the setup protocol of the blind signature scheme with input k , as well as the key generation protocol for each user $U_j \in \mathcal{U}^*$. The adversary \mathcal{B} is given all the resulting information: the public common parameters, the public and secret keys of all users in the universe.
2. **Challenge:** the adversary chooses a ring $\mathcal{U} = \{U_1, \dots, U_n\}$ of users, and two messages M_0 and M_1 . The challenger chooses at random one bit $b \in \{0, 1\}$ and initializes the interactive blind ring signature protocol with message M_b and ring \mathcal{U} as inputs. The adversary \mathcal{B} chooses some user $U_s \in \mathcal{U}$ and plays the role of the signer in the protocol (note that \mathcal{B} knows the secret key of U_s). At the end, the adversary obviously obtains \mathcal{T}_{sig} .
3. **Guess:** the adversary \mathcal{B} finally outputs its guess b' .

We say that such an adversary \mathcal{B} succeeds if $b' = b$. A scheme has the blindness property if, for all adversary \mathcal{B} , its probability of success in this game is only negligibly bigger than $1/2$.

If this probability is exactly $1/2$, for any adversary \mathcal{B} , then the blindness of the scheme is unconditional. A standard way of proving that a (ring) blind signature scheme enjoys unconditional (or perfect) blindness is by showing that the information \mathcal{T}_{sig} , that the signer obtains from an execution of the signing protocol, follows the same probability distribution for any possible message. If this is proved, then in the challenge phase of the game defined above the adversary cannot obtain from \mathcal{T}_{sig} any information about which message M_b is actually being signed, and therefore its success probability (random guess) is limited to $1/2$. This is the argument that will be used to analyze the blindness of our blind ring signature scheme.

As opposed to what is claimed by the authors of [23], where they present some “attacks” on the scheme in [12] and they consider only static groups for their scheme to avoid exactly this kind of attacks, we think that a natural definition for blindness in a blind ring signature scheme must consider only one ring of signers. Otherwise, suppose that a member of a ring executes the protocol for two pairs (m_1, \mathcal{U}_1) and (m_2, \mathcal{U}_2) of message/ring, with $\mathcal{U}_1 \neq \mathcal{U}_2$, such that he is in both rings. Later, when seeing the resulting valid signature for some of the

two messages, this signature will in particular contain the involved ring, and so he will be trivially able to distinguish which of the two passed executions was indeed the one corresponding to this message. In this way, such an adversary would break this weak notion of blindness. For this reason, we think that our definition is the good one (in particular, in step 2 of the game above, we only consider one ring and not two rings \mathcal{U}_0 and \mathcal{U}_1). This fact does not imply that a scheme with this blindness property should be used also with one ring (as suggested in [23]). The only point is that the client will only be sure that a blind signature obtained from a ring \mathcal{U} is perfectly hidden and untraceable with respect to all the blind signatures obtained from this particular ring \mathcal{U} .

3.2 Unforgeability

Unforgeability for blind ring signatures is adapted from the concept of $(\ell, \ell + 1)$ -unforgeability, introduced in [20] and maintained in [4,7] for standard blind signatures. A $(\ell, \ell + 1, q_i)$ -forger \mathcal{A} against a blind ring signature scheme is thus defined by means of the following game that it plays against a challenger:

1. **Setup:** the adversary \mathcal{A} chooses a universe \mathcal{U}^* of users and a security parameter k . The challenger runs the setup protocol of the blind signature scheme with input k , as well as the key generation protocol for each user $U_j \in \mathcal{U}^*$. It gives to the adversary \mathcal{A} the resulting common parameters and the public keys pk_j , and keeps secret the secret keys sk_j .
2. **Queries:** the forger \mathcal{A} makes different queries to the challenger:
 - q_i hash queries: if the scheme involves some hash function H_i which is assumed to behave as a random oracle [5] in the security proof, then the challenger must answer q_i queries of the adversary to this oracle, providing it with consistent and totally random values.
 - ℓ blind ring signature queries (M, \mathcal{U}) , where $\mathcal{U} \subset \mathcal{U}^*$: the challenger must answer with a valid blind ring signature Σ for this pair message/ring of users.

All these queries can be made in an adaptive way; that is, each query may depend on the answers obtained to the previous queries.

3. **Forgery:** the adversary \mathcal{A} outputs a list of $\ell + 1$ tuples $\{(M_i, \mathcal{U}_i, \Sigma_i)\}_{1 \leq i \leq \ell + 1}$. We say that \mathcal{A} *succeeds* if:
 - The $\ell + 1$ ring signatures are valid; and
 - $(M_{i_1}, \mathcal{U}_{i_1}) \neq (M_{i_2}, \mathcal{U}_{i_2})$, for all indices $1 \leq i_1, i_2 \leq \ell + 1$ such that $i_1 \neq i_2$.

Note that we require the adversary to output valid blind ring signatures for different pairs message/ring of users. That is, we do not consider as successful, for example, a forger which asks for a valid blind ring signature for the pair (M, \mathcal{U}) and later outputs as forgery two valid signatures (M, \mathcal{U}, Σ) and $(M, \mathcal{U}, \Sigma')$. Even if we do not consider, with this restriction, all the kinds of adversaries against a blind ring signature scheme, we believe that our model captures the most powerful attacks that such a scheme can suffer in practice. In effect, consider for example the application of blind ring signatures to electronic payments: a

message (a coin) is signed by a ring of banks, and later this coin is spent in some electronic transaction. The coin usually contains the date, a serial number, etc., and sellers are assumed to maintain a database with the received pairs coin/ring of banks. Therefore, an attacker which would try to spent two times the same coin, signed by the same ring of banks, should be easily detected.

4 The New Scheme

In this section we propose a blind ring signature scheme quite simple and efficient. It combines the ideas of the ring signature scheme which appears in [9] and the blind signature scheme which appears in [7]. The protocols of the new scheme are described below.

Setup and key generation. On input a security parameter k , an additive group \mathbb{G}_1 of prime order $q > 2^k$, generated by some element P , and a multiplicative group \mathbb{G}_2 with the same order q are chosen, such that they admit a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ as defined in Section 2. A hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ is also chosen. All these parameters are common and public.

Each user U_i chooses his secret key $x_i \in \mathbb{Z}_q$ at random; the matching public key is $Y_i = x_i P \in \mathbb{G}_1$.

Blind ring signature generation. The client who wants to obtain a blind ring signature on a message M with respect to a ring $\mathcal{U} = \{U_1, \dots, U_n\}$ of users, proceeds as follows: he chooses at random $r_1, \dots, r_n \in \mathbb{Z}_q$ and computes the value

$$\bar{M} = H(M, \mathcal{U}) + \sum_{i=1}^n r_i Y_i.$$

This value, along with the ring \mathcal{U} , is sent to the members of the ring. Then some of these members, say U_s , where $s \in \{1, \dots, n\}$, acts as follows:

1. For all $i \in \{1, \dots, n\}$, $i \neq s$, choose a_i uniformly at random in \mathbb{Z}_q , and compute $\bar{\sigma}_i = a_i P$.
2. Compute

$$\bar{\sigma}_s = \frac{1}{x_s} \left(\bar{M} - \sum_{i \neq s} a_i Y_i \right).$$

3. Send to the client the tuple $(\bar{\sigma}_1, \dots, \bar{\sigma}_n)$.

The client verifies if

$$e(\bar{M}, P) = \prod_{i=1}^n e(\bar{\sigma}_i, Y_i).$$

If so, he computes the values

$$\sigma_i = \bar{\sigma}_i - r_i P, \text{ for all } i = 1, \dots, n$$

and defines the signature of the message M made by the ring $\mathcal{U} = \{U_1, \dots, U_n\}$ to be $(M, \mathcal{U}, \sigma_1, \dots, \sigma_n)$.

Following the notation introduced in Section 3, we have $\mathcal{I}_C = (M, r_1, \dots, r_n)$, $\mathcal{I}_{sig} = (\mathcal{U}, \bar{M}, \{a_i\}_{i \neq s}, \bar{\sigma}_1, \dots, \bar{\sigma}_n)$ and $\mathcal{O}_C = (M, \mathcal{U}, \Sigma)$, where $\Sigma = (\sigma_1, \dots, \sigma_n)$.

Verification of a blind ring signature. The validity of the signature $(M, \mathcal{U}, \sigma_1, \dots, \sigma_n)$ is verified by checking if

$$e(H(M, \mathcal{U}), P) = \prod_{i=1}^n e(\sigma_i, Y_i).$$

Correctness and anonymity of the resulting scheme directly infer from the properties satisfied by the aforementioned schemes in [9,7]. In particular, the anonymity property holds unconditionally: even if a client has unlimited computational resources (which means for example that he can obtain the secret keys of all the members of a ring) he cannot obtain any information about which member has actually participated in the interactive protocol to compute a blind ring signature.

Note that unconditional anonymity directly implies a different property, *unlinkability* [16], which means that nobody (including the client) will be able to distinguish if two different interactive executions of the blind ring signature protocol have been performed by the same member of the ring or not. In effect, if a scheme is linkable, then there exists a polynomial-time linking algorithm which takes as input two executions of the blind ring signature protocol and outputs 1 if and only if the same member of the ring has participated in both executions. If this holds, then a client with unlimited resources who tries to break the anonymity of some execution of the protocol can act as follows: (1) he obtains all the secret keys of the members of the ring; (2) for each member U_i of the ring, the client uses the obtained secret key to run by himself a new interactive execution of the blind ring signature protocol; (3) the client applies the linking algorithm to this last execution and to the initial execution whose anonymity he is trying to break; (4) if the output of the linking algorithm is 1 for user U_i , then this user was the one who participated in the initial (target) execution.

We now prove that the scheme also satisfies the properties of blindness and unforgeability.

4.1 Blindness of the Scheme

As stated in Section 3.1, we can prove that the proposed scheme achieves unconditional blindness if we prove that the probability distribution of the information \mathcal{T}_{sig} that the signer (the adversary in the blindness game) obtains in an execution of the signing protocol is exactly the same for any possible message. In the case of our scheme, we have $\mathcal{T}_{sig} = (\mathcal{U}, \bar{M}, \{a_i\}_{i \neq s}, \bar{\sigma}_1, \dots, \bar{\sigma}_n)$, where $U_s \in \mathcal{U} = \{U_1, \dots, U_n\}$ is a user chosen by the adversary.

The value $\bar{M} = H(M, \mathcal{U}) + \sum_{i=1}^n r_i Y_i$ follows a completely random and uniform distribution in \mathbb{G}_1 , independently of the message M , because all the integers $r_i \in \mathbb{Z}_q$ are chosen uniformly and at random. For the rest of values in \mathcal{T}_{sig} , either they are chosen by the adversary or they depend on \bar{M} . In any case, their probability distribution does not depend on the signed message M .

Summing up, during the challenge phase of the blindness game (see Section 3.1), the information that the adversary obtains if the challenger chooses M_0

is perfectly indistinguishable from the information that the adversary obtains if the challenger chooses M_1 . Therefore, the scheme achieves perfect blindness.

4.2 Unforgeability of the Scheme

We are going to prove that our scheme is $(\ell, \ell + 1)$ -unforgeable in the random oracle model, and under the assumption that the Chosen-Target-Inverse-CDH problem is hard to solve. We denote as q_1 the number of queries that an adversary \mathcal{A} against the unforgeability of our scheme can make to the (random) oracle which models the behaviour of the hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$.

Theorem 1. *If there exists a $(\ell, \ell + 1, q_1)$ -forger \mathcal{A} against the unforgeability of our blind ring signature scheme, which succeeds with probability ε , then there exists a (q_t, q_h, d) -solver \mathcal{S}' of the Chosen-Target-Inverse-CDH problem, which also succeeds with probability $\varepsilon' \geq \varepsilon - \frac{\ell+1}{q}$, where q is the order of the group \mathbb{G}_1 , $q_t = d$, $d = \ell + 1$ and $q_h = \ell$.*

Proof. Assuming the existence of such a forger \mathcal{A} , let us construct a solver \mathcal{S}' of the Chosen-Target-Inverse-CDH problem. First of all, \mathcal{S}' initializes \mathcal{A} , which chooses a security parameter k and a universe of users \mathcal{U}^* . Solver \mathcal{S}' chooses a group \mathbb{G}_1 with primer order $q > 2^k$ which admits a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

After that, solver \mathcal{S}' asks for an instance of the Chosen-Target-Inverse-CDH problem in the group \mathbb{G}_1 . It receives a pair (P', Y') , where $Y' = a'P'$ for some random and secret value $a' \in \mathbb{Z}_q$; it is also provided with access to the target and the helper oracles.

For each user $U_j \in \mathcal{U}^*$, solver \mathcal{S}' defines his public key to be $Y_j = \alpha_j Y'$, for some random value $\alpha_j \in \mathbb{Z}_q^*$. At this point, \mathcal{S}' sends to \mathcal{A} all the common parameters q , $\mathbb{G}_1 = \langle P' \rangle$, \mathbb{G}_2 , e , the public keys Y_j of all the users U_j in the universe, and provides it with access to a random oracle for a hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$.

Hash queries: the forger \mathcal{A} makes q_h queries $\mathcal{Q}_i = (M_i, \mathcal{U}_i)$ to the random oracle. Solver \mathcal{S}' maintains a table TAB where it stores the relations $H(\mathcal{Q}_i) = Z_i$ that it computes as follows: if a received query $\mathcal{Q}_i = (M_i, \mathcal{U}_i)$ is already in the table, \mathcal{S}' sends to \mathcal{A} the stored value Z_i . If not, \mathcal{S}' makes a query to its target oracle; it receives as answer a random element $Z_i \in \mathbb{G}_1$. Then it stores the new relation $H(\mathcal{Q}_i) = Z_i$ in TAB and sends Z_i to the forger \mathcal{A} .

Blind ring signature queries: the forger \mathcal{A} is assumed to initialize ℓ times the interactive blind ring signature protocol, playing the role of the client. Solver \mathcal{S}' must play the role of the signers and simulate the information that \mathcal{A} should obtain in a real execution of this protocol. The forger \mathcal{A} sends a message $\bar{M} \in \mathbb{G}_1$ to be signed by a ring $\mathcal{U} = \{U_1, \dots, U_n\}$. Then solver \mathcal{S}' acts as follows:

1. it chooses at random a user $U_s \in \mathcal{U}$. For $i \in \{1, \dots, n\}$, $i \neq s$, the solver chooses random values $a_i \in \mathbb{Z}_q$ and computes $\bar{\sigma}_i = a_i P'$;

2. it sends to the helper oracle the value

$$W_i = \frac{1}{\alpha_s} \left(\bar{M} - \sum_{i \neq s} a_i Y_i \right),$$

and obtains as answer the value $\bar{\sigma}_s = \frac{1}{a'} W_i$;

3. \mathcal{S}' sends to \mathcal{A} the tuple $(\bar{\sigma}_1, \dots, \bar{\sigma}_n)$.

In effect, this tuple perfectly simulates the information that \mathcal{A} would have obtained in a real execution of the protocol, since

$$\begin{aligned} \prod_{i=1}^n e(\bar{\sigma}_i, Y_i) &= e(\bar{\sigma}_s, Y_s) \prod_{i \neq s} e(a_i P', Y_i) = \\ &= e \left(\frac{1}{a' \alpha_s} \left(\bar{M} - \sum_{i \neq s} a_i Y_i \right), \alpha_s a' P' \right) \prod_{i \neq s} e(a_i P', Y_i) = \\ &= e(\bar{M}, P') \prod_{i \neq s} e(-a_i Y_i, P') e(a_i P', Y_i) = e(\bar{M}, P'). \end{aligned}$$

The environment of \mathcal{A} is thus perfectly simulated by \mathcal{S}' , so with probability ε the forger \mathcal{A} outputs $\ell + 1$ tuples $\{(M_i, \mathcal{U}_i, \Sigma_i)\}_{1 \leq i \leq \ell+1}$ of valid ring signatures such that all the pairs (M_i, \mathcal{U}_i) in these tuples are different. Since the hash function H is assumed to behave as a random function, the probability that \mathcal{A} obtains a valid ring signature for (M_i, \mathcal{U}_i) without asking for the value $H(M_i, \mathcal{U}_i)$ is $1/q$. Therefore, we have that with probability $1 - \frac{\ell+1}{q}$ the forger \mathcal{A} has queried the random oracle with (M_i, \mathcal{U}_i) , for the $\ell + 1$ forged pairs. This means that, for $i = 1, \dots, \ell + 1$, we have that $H(M_i, \mathcal{U}_i) = Z_{j_i}$ where Z_{j_i} are elements given to \mathcal{S}' by its target oracle. The signatures are valid, so

$$e(H(M_i, \mathcal{U}_i), P') = \prod_{U_j \in \mathcal{U}_i} e(\sigma_{ij}, Y_j) = e \left(\sum_{U_j \in \mathcal{U}_i} \alpha_j a' \sigma_{ij}, P' \right)$$

For $i = 1, \dots, \ell + 1$, solver \mathcal{S}' outputs the pair (V_i, j_i) , where

$$V_i = \sum_{U_j \in \mathcal{U}_i} \alpha_j \sigma_{ij}$$

satisfies $V_i = \frac{1}{a'} H(M_i, \mathcal{U}_i) = \frac{1}{a'} Z_{j_i}$, as desired. Furthermore, since all the pairs (M_i, \mathcal{U}_i) are assumed to be different, we have that all the values V_i are also different.

Summing up, solver \mathcal{S}' makes $q_t \leq q_1$ queries to its target oracle, makes $q_h = \ell$ queries to its helper oracle, and with probability $\varepsilon' \geq \varepsilon - \frac{\ell+1}{q}$ outputs $d = \ell + 1$ valid pairs (V_i, j_i) . \square

5 Conclusions

We proposed a simple and quite efficient pairing-based ring signature scheme. It is based on Boneh *et al.* ring signatures and on Boldyreva's blind signature, and naturally inherits the advantages and drawbacks of both constructions: the number of scalar multiplications to compute a signature grows linearly with the number of members in the ring, as well as the number of pairing evaluations for the verification, and the size of the signature itself. The scheme remains practical anyway, for rings of reasonable size. Furthermore, it achieves unconditional blindness and anonymity, as opposed to previous blind ring signature schemes. Unforgeability of the scheme is proved in the random oracle, under some quite standard assumptions.

An open problem would be to build a practical scheme whose unforgeability could be proved in the standard model. Blind signatures and ring signatures without random oracles have been recently proposed [10,6], so maybe it is possible to combine them and obtain blind ring signatures in the standard model. Another open question deals with the possibility of modifying our scheme so that the size of the signatures becomes constant, independent of the number of signers in the ring. A possible strategy to achieve this could be the use of accumulators based on pairings [18].

References

1. Advances in elliptic curve cryptography, edited by I. F. Blake, G. Seroussi and N. P. Smart, LMS **317**, Cambridge University Press (2005).
2. G. Ateniese, J. Camenisch, M. Joye and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. *Proceedings of Crypto'00*, LNCS **1880**, Springer-Verlag, pp. 255–270 (2000).
3. M. Bellare, D. Micciancio and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. *Proceedings of Eurocrypt'03*, LNCS **2656**, Springer-Verlag, pp. 614–629 (2003).
4. M. Bellare, C. Namprepmpre, D. Pointcheval and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology*, Vol. **16** (3), Springer-Verlag, pp. 185–215 (2003).
5. M. Bellare and P. Rogaway. Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. *Proceedings of CCS'93*, ACM Academic Press, pp. 62–73 (1993).
6. A. Bender and J. Katz and R. Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. *Proceedings of TCC'06*, LNCS **3876**, Springer-Verlag, pp. 60–79 (2006).
7. A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. *Proceedings of PKC'03*, LNCS **2567**, Springer-Verlag, pp. 31–46 (2003).
8. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. *Proceedings of Crypto'04*, LNCS **3152**, Springer-Verlag, pp. 41–55 (2004).

9. D. Boneh, C. Gentry, B. Lynn and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. *Proceedings of Eurocrypt'03*, LNCS **2656**, Springer-Verlag, pp. 416–432 (2003).
10. J. Camenisch, M. Koprowski and B. Warinschi. Efficient Blind Signatures Without Random Oracles. *Proceedings of SCN'04*, LNCS **3352**, Springer-Verlag, pp. 134–148 (2005).
11. J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. *Proceedings of Crypto'97*, LNCS **1294**, Springer-Verlag, pp. 410–424 (1997).
12. T. K. Chan, K. Fung, J. K. Liu and V. K. Wei. Blind Spontaneous Anonymous Group Signatures for Ad Hoc Groups. *Proceedings of ESAS'04*, LNCS **3313**, Springer-Verlag, pp. 82–84 (2005).
13. D. Chaum. Blind Signatures for Untraceable Payments. *Proceedings of Crypto'82*, pp. 199–203 (1982).
14. D. Chaum, E. van Heyst. Group Signatures. *Proceedings of Eurocrypt'91*, LNCS **547**, Springer-Verlag, pp. 257–265 (1991).
15. Y. Dodis, A. Kiayias, A. Nicolosi and V. Shoup. Anonymous Identification in Ad-Hoc Groups. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 609–626 (2004).
16. J.K. Liu, V.K. Wei and D.S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. *Proceedings of ACISP'04*, LNCS **3108**, Springer-Verlag, pp. 325–335 (2004).
17. A. Lysyanskaya and Z. Ramzan. Group Blind Digital Signatures: a Scalable Solution to Electronic Cash. *Proceedings of FC'98*, LNCS **1465**, Springer-Verlag, pp. 184–197 (1998).
18. L. Nguyen. Accumulators from Bilinear Pairings and Applications. *Proceedings of CT-RSA'05*, LNCS **3376**, Springer-Verlag, pp. 275–292 (2005).
19. K. Q. Nguyen, Y. Mu and V. Varadharajan. Divertible Zero-Knowledge Proof of Polynominal Relations and Blind Group Signature. *Proceedings of ACISP'99*, LNCS **1587**, Springer-Verlag, pp. 117–128 (1999).
20. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, Vol. **13** (3), Springer-Verlag, pp. 361–396 (2000).
21. R. L. Rivest, A. Shamir and Y. Tauman. How to Leak a Secret. *Proceedings of Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 552–565 (2001)
22. G. Wang. Bibliography on Digital Signatures. <http://www.i2r.a-star.edu.sg/icsd/staff/guilin/bible.htm>
23. Q. Wu, F. Zhang, W. Susilo and Y. Mu. An Efficient Static Blind Ring Signature Scheme. *Proceedings of ICISC'05*, LNCS, Springer-Verlag, to appear (2006).

Multi-party Concurrent Signatures

Dongvu Tonien, Willy Susilo, and Reihaneh Safavi-Naini

School of Information Technology and Computer Science
University of Wollongong, Australia
{dong, wsusilo, rei}@uow.edu.au

Abstract. The concept of concurrent signatures was introduced by Chen, Kudla and Paterson at Eurocrypt 2004. In a concurrent signature scheme, users sign their messages in an ambiguous way so that the signatures are only verifiable by the users themselves *but not* by any other outsiders. At a later stage, one of the users releases an extra bit of information called the *keystone*, then all the signatures become binding to their signers concurrently. At this stage, any outsider can verify the signatures. Chen, Kudla and Paterson proposed a concurrent signature scheme for *two* users. Recently, Susilo and Mu constructed a scheme for *three* users. It is an open problem to construct concurrent signature schemes for multi users. In this paper, we answer this open problem affirmatively. Using techniques of ring signatures and bilinear pairings, *for the first time* we construct a concurrent signature scheme for multi-users.

Keywords: concurrent signature, ring signature, bilinear pairings.

1 Introduction

Fair exchange in digital signatures is one of the fundamental problems in cryptography. Fair exchange is a necessary feature in many applications for electronic commerce. One of such applications is contract signing protocol where a number of parties need to exchange their signatures on a contract.

Two party fair exchange has been studied extensively in the literature [12,11,2,13,3,4,5,10,17]. One of the techniques [12,11,4,14] is based on the idea of timed release or timed fair exchange of signatures in which two parties interactively exchange their signatures “little-by-little”. The other technique [2,13,3,5,10,17] involves the use of a semi-trusted third party or arbitrator, who can be called upon handle disputes between signers.

In 2004, Chen, Kudla and Paterson [6] introduce a weaker version of two party fair exchange called *concurrent signature* that does not require any third trusted party and does not require many interactive exchange protocols. In this two-party concurrent signature, two signers produce two signatures in such a way that from any outsider’s point of view, both signatures are ambiguous. And after an additional information called *keystone* is released by one of the parties, both signatures are binding concurrently.

Chen, Kudla and Paterson questioned the existence of multi-party concurrent signatures. They noted that if multi-party concurrent signatures can be constructed

and modelled correctly, this will move closer to the full solution of multi-party fair exchange problem. Recently, Susilo and Mu [19] have successfully constructed a three-party concurrent signature scheme. However, their solution as well as Chen *et al.* solution seem difficult to be extended to the multi-party case.

Our contribution. In this paper, we propose *for the first time* a multi-party concurrent signature scheme, thus, solving affirmatively the open problem questioned by Chen *et al.* and Susilo *et al.* The main ingredient in our construction is a ring signature scheme. Ring signature was introduced by Rivest, Shamir and Tauman [18] in 2001. Without the help of any trusted third party, it allows a user to sign a signature for a ring of users in an anonymous way. From the point of view of an outsider, the signature could have been generated by anyone in the ring. Our multi-party concurrent signature scheme is based particularly on a recently proposed ring signature by Chow, Yiu and Hui [8].

To extend to multi-user scenario, we introduce a new model for using keystones. In previous models, only a *single keystone* selected by the initial user is used in the signing phase. This gives the initial user a decisive power over the other users that the initial user can throw the keystone at anytime even when some of the users have not signed their messages. In our model, the initial user chooses a *group keystone* and additionally, all users can select their own *individual keystones*. Thus, the binding of the signatures happens after all users release their keystones.

Related works. Concurrent signature schemes remove the requirement of a dispute-resolving TTP in fair exchange of signatures; however, it still requires a CA, like a normal signature scheme. This notion has been extended in [20] to a new concept called *perfect concurrent signatures*, with the aim to further anonymize the signatures before the keystone is released. In this concept, even the signers are known to be trustworthy, the signatures remain anonymous to the third party before the keystone is released. The generic construction of perfect concurrent signatures was recently proposed in [7].

Designated verifier proofs were proposed in [15]. The idea is to allow signatures to convince only the intended recipient, who is assumed to have a public-key. As noted in [18], ring signature schemes can be used to provide this mechanism by joining the verifier in the ring. However, it might not be practical in real life since the verifier might not have any public key setup. As noted in [6], intuitively concurrent signatures can be built from either ring signature schemes or designated verifier proofs, but [6] also noted that construction from designated verifier proofs is not straightforward and it cannot achieve the required properties of concurrent signatures.

The paper is organized as follows. In Section 2, we present the formal model of a multi-party concurrent signature scheme and its security consideration. We also introduce bilinear pairings and related complexity assumptions. In Section 3, we present our multi-party concurrent signature scheme which is based on a recently proposed Chow-Yiu-Hui's ring signature scheme, together with its security proof. Section 4 concludes the paper.

2 Definitions

In this section, we give the description of a multi-party concurrent signature scheme. Main algorithms in a multi-party concurrent signature scheme are defined in Section 2.1. Section 2.2 shows how these algorithms are used in the multi-party concurrent signature scheme. Security requirements for concurrent signatures are discussed in Section 2.3. We briefly introduce bilinear pairings and their related complexity assumptions in Section 2.4.

Let there be n users. We will call these n users *the insiders* to distinguish them with any other parties as *the outsiders*. Intuitively, multi-party concurrent signature schemes work as follows.

One of the users will initiate the signing of concurrent signatures. Let us call this user the *initial signer*. The initial signer U_1 chooses a *group keystone* and an *individual keystone* and keeps them secret. The group keystone is hashed to a *group keystone-fix* by a public hash function. U_1 uses the individual keystone together with the group keystone-fix to produce an anonymous signature. U_1 then sends the first ambiguous signature together with the group keystone-fix to all users. This signature can be verified in two different levels by the outsiders and the insiders, respectively. The outsiders can verify the signature as an anonymous signature but cannot link the signature to any specific user. In the other words, the outsiders can be sure that the signature is created by one of the users but cannot identify the signer of the signature. The insiders, on the other hand, can verify the signature in a deeper level. With their secret keys, the insiders can verify the signature as a true signature of the initial user U_1 .

After the insiders have all verified U_1 's signature, another user, say U_2 , responds to U_1 . User U_2 then chooses a *second individual keystone*, keeps it secret, and uses it together with the group keystone-fix to produce a second ambiguous signature. Again, this signature of U_2 is sent to all users and can be verified differently on two levels by the outsiders and the insiders as the first signature. Proceeding similarly, eventually, after all users have signed their messages, the *keystone releasing phase* starts. In this phase, all users release their individual keystones. The initial user also releases the group keystone. With these released keystones, all signatures bind to their signers concurrently. At this point, any outsider is able to use the keystones to verify all signatures to their signers (and hence, the signatures are valid from their point of view as well).

We should remark a crucial difference in our model with the previous models [6,19] on the uses of keystones. In previous models, only a *single keystone* is used in the signing phase. This gives the initial user a decisive power over the other users. The initial user in these models can maliciously releases the keystone anytime he/she wants even when some of the users have not signed their messages. In our model, by allowing each user chooses his/her own individual keystone, we distribute the keystone-releasing right evenly to *all* users. We think that this model is a natural generalization of the previous model in multi-user setting. One may wonder what would happen if some users decide to release their keystones before the signing phase completes. In this case, these users dispose their identities and have their signatures linked to them, while

the other users who have not released the keystones remain anonymous with their signatures. So the single keystone in the previous models corresponds to what is now called the group keystone in our model. The group keystone is hashed to a keystone-fix and initially sent to all users so that all users can produce their signatures consistent and synchronized with each other via the keystone-fix.

We also remark that the users do *not* necessarily sign their signatures on the same message. Following the previous models, we allow each user to sign signature on his/her own arbitrary message.

2.1 Multi-party Concurrent Signature Algorithms

A multi-party concurrent signature scheme consists of the following main algorithms:

- **Setup**(ℓ, n): On input ℓ – a security parameter, n – number of users, the algorithm **Setup** outputs n user secret keys $SK = \{sk_i\}_{i=1}^n$, a public key PK , the descriptions of a message space \mathcal{M} , a keystone space \mathcal{K} , a keystone-fix space \mathcal{F} , other system public parameter **params**, and a function $\text{KGEN} : \mathcal{K} \rightarrow \mathcal{F}$;
- **Sign**(m, PK, F, sk, ik): On input a message m , the public key tuple PK , a keystone-fix F , a user's secret key sk , and an individual keystone ik , the algorithm **Sign** outputs an anonymous signature σ for m . This algorithm is used by an *insider* to generate an anonymous signature;
- **OutsiderVerify**(m, σ, PK, F): On input a message m , a signature σ , the public key tuple PK , a keystone-fix F , the algorithm **OutsiderVerify** outputs either **accept** or **reject**. This algorithm is used by an *outsider* to check an anonymous signature;
- **InsiderVerify**(m, σ, PK, F, j, sk): On input a message m , a signature σ , the public key tuple PK , a keystone-fix F , a user identity j , and a user's secret key sk , the algorithm **InsiderVerify** outputs either **accept** or **reject**. This algorithm is used by an *insider* (with a secret key sk) to check whether the user j has signed the signature or not;
- **KeystonesRelease**: After the initial user releases the group keystone gk and his/her individual keystone ik_1 , and other users release their individual keystones ik_i , the algorithm **KeystonesRelease** outputs the final binding signature in an agreed format $\langle gk, F, ik_1, \dots, ik_n, m_1, \sigma_1, m_2, \sigma_2, \dots, m_n, \sigma_n \rangle$, where (m_i, σ_i) is the signature of user i ;
- **BindingVerify**($PK, \langle gk, F, ik_1, \dots, ik_n, m_1, \sigma_1, m_2, \sigma_2, \dots, m_n, \sigma_n \rangle$): On input the public key tuple PK and a concurrent signature

$$\langle gk, F, ik_1, \dots, ik_n, m_1, \sigma_1, m_2, \sigma_2, \dots, m_n, \sigma_n \rangle,$$

the algorithm **BindingVerify** outputs either **accept** or **reject**. This algorithm is used by anyone to check the validity and binding of a list of n signatures.

2.2 Multi-party Concurrent Signature Scheme – How Does it Work?

A multi-party concurrent signature scheme works as follows

1. All users get together and run **Setup** algorithm, which gives them all system public parameters and their secret keys.
2. An initial user, U_1 , initiates a concurrent signature signing. U_1 picks a random group keystone gk and an individual keystone ik_1 . U_1 computes the group keystone-fix $F = \text{KGEN}(gk)$. With the keystone-fix F , the individual keystone ik_1 , the public key tuple PK , and U_1 's secret key sk_1 , user U_1 then uses the **Sign** algorithm to sign a message m_1 , which produces an anonymous signature $\sigma_1 = \text{Sign}(m_1, PK, F, sk_1, ik_1)$. User U_1 keeps the group keystone gk and the individual keystone ik_1 secret and sends (m_1, σ_1, F) to all other users;
3. Upon receiving U_1 signature (m_1, σ_1) and the group keystone-fix F , each user U_j uses its secret key sk_j , executes the algorithm

$$\text{InsiderVerify}(m_1, \sigma_1, PK, F, U_1, sk_j)$$

to verify whether the signature is valid from user U_1 ;

4. If no users reject U_1 signature, another user U_2 may response to U_1 . User U_2 picks an individual keystone ik_2 . With the group keystone-fix F , the individual keystone ik_2 , the public key tuple PK , the secret key sk_2 , user U_2 calls the **Sign** algorithm to sign a message m_2 , which produces an anonymous signature $\sigma_2 = \text{Sign}(m_2, PK, F, sk_2, ik_2)$. User U_2 then sends (m_2, σ_2) to all other users;
5. If no users reject U_2 signature, another user U_3 may response to U_2 and sends (m_3, σ_3) to all other users. This process is repeated until eventually all users sign their messages and send their signatures to other users;
6. After all users sign their messages, the initial user U_1 releases the group keystone gk and his/her individual keystone ik_1 . Other user U_j follows to release his/her corresponding individual keystone ik_j . The algorithm **KeystonesRelease** is executed to output the concurrent signature in its final form. From this point, anyone can execute the **BindingVerify** algorithm

$$\text{BindingVerify}(PK, \langle gk, F, ik_1, \dots, ik_n, m_1, \sigma_1, m_2, \sigma_2, \dots, m_n, \sigma_n \rangle)$$

to verify the concurrent signature;

7. Note that any *outsider* can execute **OutsiderVerify** (m_i, σ_i, PK, F) in steps 3, 4, and 5 to check the validity of the signature (m_i, σ_i) that indeed it is generated by one of the users. However, this signature still remains anonymous and an outsider cannot prove that U_i has signed (m_i, σ_i) .

Remark

- Even it is not stated explicitly in the scheme description, we assume that each time a user chooses a keystone, the user must pick a new keystone that has not been used before.

- There are two verification algorithms that can be executed by the outsiders. The first algorithm `OutsiderVerify` is to be executed *before* the release of keystones and the outsiders *do not* know the identity of the signature owner. The second algorithm `BindingVerify` is to be executed *after* the release of keystones and the outsiders *do* know which user has signed which individual signature.

2.3 Security Requirements

We require a multi-party concurrent signature scheme to satisfy four properties: *correctness*, *unforgeability*, *ambiguity* and *fairness*. Intuitively, these notions are described as follows.

- *Correctness*: if a signature has been generated correctly by invoking `Sign` algorithm, then checking this signature, the two algorithms `InsiderVerify` and `OutsiderVerify` will output *accept*. Moreover, after a keystone is released, the algorithm `BindingVerify` also outputs *accept* on this signature.
- *Unforgeability*: there are two different cases that we need to consider
 - *Outside Attacker*: when an adversary does not have users' secret keys, then no valid signature that will pass the `OutsiderVerify` algorithm can be produced;
 - *Inside Attacker*: for any two users u and v , any group of other users cannot frame user v by producing a signature that makes user u believe that the signature is created by user v when user u executes the algorithm `InsiderVerify`.
- *Ambiguity*: before the individual keystone ik_j is released, an adversary cannot prove that user j is the actual signer of a signature. Instead, the only thing that the adversary can be sure is that the actual signer belongs to the group of n users.
- *Fairness*: We require that any valid ambiguous signatures generated using the same group keystone-fix will all become binding after the keystones are released. Hence, a matching signer cannot be left in a position where a keystone binds his signature to him whilst some of the other signers' signatures are not bound to them.

2.4 Bilinear Pairings and Complexity Assumptions

Let G_1 and G_2 be two groups of the same prime order q . Normally, G_1 is a group of elliptic curve points and G_2 is a subgroup of a group of non-zero elements of a fields. We will write group operation in G_1 additively and in G_2 multiplicatively. Elements of G_1 are usually presented as capital letters like P, Q, S, \dots , and elements of G_2 are presented as lower case letters.

A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the following three properties

1. *Bilinearity*: for any $\alpha, \beta \in \mathbf{Z}_q$ and $P, Q \in G_1$,

$$\hat{e}(\alpha P, \beta Q) = \hat{e}(P, Q)^{\alpha\beta};$$

2. *Non-degeneracy*: for all $P \in G_1$, if $P \neq \mathcal{O}$ then $\hat{e}(P, P) \neq 0$;
3. *Computability*: there exists an efficient algorithm to calculate $\hat{e}(P, Q)$ for all $P, Q \in G_1$.

We consider the following bilinear Diffie–Hellman assumption.

Computational Bilinear Diffie–Hellman Assumption. The bilinear Diffie–Hellman problem in (G_1, G_2, \hat{e}) is as follows: given (P, xP, yP, zP) for some $x, y, z \in \mathbf{Z}_q^*$, compute $g = \hat{e}(P, P)^{xyz} \in G_2$. The computational bilinear Diffie–Hellman assumption over (G_1, G_2, \hat{e}) states that any poly-time algorithm \mathcal{A} has negligible success in solving the bilinear Diffie–Hellman problem, that is,

$$\Pr[\mathcal{A}(P, xP, yP, zP) = \hat{e}(P, P)^{xyz}]$$

is negligible in $\log q$, where the probability is over the random choice of $x, y, z \in \mathbf{Z}_q^*$ and the random choice $P \in G_1$.

Computational Diffie–Hellman Assumption on G_1 . The Diffie–Hellman problem in G_1 is as follows: given (P, xP, yP) for some $x, y \in \mathbf{Z}_q^*$, compute $xyP \in G_1$. The computational Diffie–Hellman assumption over G_1 states that any poly-time algorithm \mathcal{A} has negligible success in solving the Diffie–Hellman problem, that is,

$$\Pr[\mathcal{A}(P, xP, yP) = xyP]$$

is negligible in $\log q$, where the probability is over the random choice of $x, y \in \mathbf{Z}_q^*$ and the random choice $P \in G_1$.

Relationship between the two assumptions

Clearly, if from P, xP, yP we can calculate xyP then with zP we can calculate $\hat{e}(P, P)^{xyz} = \hat{e}(zP, xyP)$. Thus the Computational Bilinear Diffie–Hellman Assumption is stronger than the Computational Diffie–Hellman Assumption.

3 The Proposed Multi-party Concurrent Signatures

In this section we will describe our multi-party concurrent signature scheme. Our construction is inspired by the recently proposed ring signature scheme of Chow, Yiu and Hui [8].

– Algorithm Setup(ℓ, n):

- Select bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ where G_1 and G_2 are groups of prime order q of ℓ bits, and four cryptographic hash functions

$$H_1, H_2 : \{0, 1\}^* \rightarrow G_1 \text{ and } H_3, H_4 : \{0, 1\}^* \rightarrow \mathbf{Z}_q;$$

- Select a generator P of G_1 and random numbers $sk_0, sk_1, \dots, sk_n \in \mathbf{Z}_q^*$. For each $0 \leq i \leq n$, set $pk_i = P_i = sk_i P$;
- For each $i \in [1, n]$, the secret key for user i is sk_i ;
- The public key tuple is $PK = (P, P_0, P_1, P_2, \dots, P_n)$;
- Set $\mathcal{K} = \{0, 1\}^*$, $\mathcal{F} = G_1$, $\mathcal{M} = \mathbf{Z}_q$;

- The function $\text{KGEN} : \mathcal{K} \rightarrow \mathcal{F}$ is defined as $\text{KGEN} = H_1$.

– **Algorithm** $\text{Sign}(m, PK, F, sk, ik)$:

A user j uses a secret key $sk = sk_j$ and an individual keystone $ik = ik_j$ to sign a message $m = m_j$ as follows,

- For each $i = 1, \dots, n$ such that $i \neq j$, compute

$$U_i = H_2(m_j || j || F || \hat{e}(P_0, P_i)^{sk_j}) \in G_1$$

and

$$h_i = H_3(m_j || i || PK || F || U_i) \in \mathbf{Z}_q;$$

- Compute

$$U_j = H_4(ik_j)P_j - \sum_{i \neq j} (U_i + h_i P_i) \in G_1,$$

$$h_j = H_3(m_j || j || PK || F || U_j) \in \mathbf{Z}_q$$

and

$$V = sk_j(h_j + H_4(ik_j))F \in G_1;$$

- The signature is $\sigma = (V, U_1, U_2, \dots, U_n)$.

– **Algorithm** $\text{OutsiderVerify}(m, \sigma, PK, F)$:

- For each $i = 1, 2, \dots, n$, compute $h_i = H_3(m || i || PK || F || U_i) \in \mathbf{Z}_q$;
- Outputs accept if $\hat{e}(P, V) = \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i))$.

– **Algorithm** $\text{InsiderVerify}(m, \sigma, PK, F, j, sk)$:

A user i uses a secret key $sk = sk_i$ to check signature σ on a message $m = m_j$ from user j as follows,

- If $U_i \neq H_2(m_j || j || F || \hat{e}(P_0, P_j)^{sk_i})$ then outputs reject, otherwise continue;
- For each $i = 1, 2, \dots, n$, compute $h_i = H_3(m_j || i || PK || F || U_i) \in \mathbf{Z}_q$;
- Outputs accept if $\hat{e}(P, V) = \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i))$.

– **Algorithm** $\text{BindingVerify}(PK, \langle gk, F, ik_1, ik_2, \dots, ik_n, r_1, r_2, \dots, r_n, m_1, \sigma_1, m_2, \sigma_2, \dots, m_n, \sigma_n \rangle)$: consequently check the following conditions, outputs reject if any condition fails,

- Check if $F \stackrel{?}{=} \text{KGEN}(gk)$ holds;
- For each $j = 1, 2, \dots, n$, in signature from user j , check if

$$\hat{e}(P, V) \stackrel{?}{=} \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i)),$$

and

$$\hat{e}(P, V) \stackrel{?}{=} \hat{e}(P_j, (h_j + H_4(ik_j))F),$$

where $h_i = H_3(m_j || i || PK || F || U_i) \in \mathbf{Z}_q$.

Remarks

- The difference between algorithms `OutsiderVerify` and `InsiderVerify` is that, the algorithm `InsiderVerify` requires an additional checking

$$U_i \stackrel{?}{=} H_2(m_j || j || F || \hat{e}(P_0, P_j)^{sk_i}).$$

This can be executed by user i who has the secret key sk_i , but cannot be executed by any outsider.

- The difference between algorithms `OutsiderVerify` and `BindingVerify` is that, the algorithm `BindingVerify` requires additional checkings

$$F \stackrel{?}{=} KGEN(gk) \quad \text{and} \quad \hat{e}(P, V) \stackrel{?}{=} \hat{e}(P_j, (h_j + H_4(ik_j))F).$$

This can only be executed when the group keystone gk and the individual keystone ik_j are known.

- *Efficiency*: Since we allow each user to sign signature on his/her own arbitrary message, for n user, a final concurrent signature size must be n times the size of each individual signature. Our proposed concurrent signature scheme is based on a linear-size ring signature [8], thus, each individual signature is of size $\mathcal{O}(n)$, and the final concurrent signature is of size $\mathcal{O}(n^2)$. It is an open problem to construct a concurrent signature scheme based on a constant-size ring signature. If constant-size ring signatures [9,16] can be used then the resulted concurrent signature would have the size reduced to $\mathcal{O}(n)$.

3.1 Security Analysis

Correctness. If a signature has been generated correctly by invoking `Sign` algorithm then it should pass all three algorithms `InsiderVerify`, `OutsiderVerify` and `BindingVerify`.

Theorem 1. *The proposed scheme satisfies the correctness property.*

Proof

1. Checking algorithm `OutsiderVerify`:

Since $U_j = H_4(ik_j)P_j - \sum_{i \neq j} (U_i + h_i P_i)$ we have

$$\sum_{i \neq j} (U_i + h_i P_i) = H_4(ik_j)P_j - U_j,$$

thus,

$$\sum_{i=1}^n (U_i + h_i P_i) = (H_4(ik_j)P_j - U_j) + (U_j + h_j P_j) = (H_4(ik_j) + h_j)P_j.$$

Therefore,

$$\hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i)) = \hat{e}(F, (H_4(ik_j) + h_j)P_j) = \hat{e}(F, sk_j(H_4(ik_j) + h_j)P).$$

On the other hand, $V = sk_j(h_j + H_4(ik_j))F$. Thus,

$$\hat{e}(P, V) = \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i)),$$

and the signature passes the algorithm `InsiderVerify`.

2. Checking algorithm `InsiderVerify`

Since $\hat{e}(P_0, P_i)^{sk_j} = \hat{e}(P, P)^{sk_0 sk_i sk_j} = \hat{e}(P_0, P_j)^{sk_i}$, we have

$$U_i = H_2(m_j || j || F || (P_0, P_i)^{sk_j}) = H_2(m_j || j || F || (P_0, P_j)^{sk_i}).$$

In addition, as shown above, $\hat{e}(P, V) = \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i))$, thus, the signature passes the algorithm `InsiderVerify`.

3. Checking algorithm `BindingVerify`

Since $V = sk_j(h_j + H_4(ik_j))F$, we have

$$\hat{e}(P, V) = \hat{e}(P, sk_j(h_j + H_4(ik_j))F) = \hat{e}(P_j, (h_j + H_4(ik_j))F).$$

In addition, as shown above, $\hat{e}(P, V) = \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i))$, thus, the signature passes the algorithm `BindingVerify`. This completes the proof. ■

Unforgeability

Outside Attacker: when an adversary does not have users' secret keys, then no valid signature that will pass the `OutsiderVerify` algorithm can be produced.

Theorem 2. (Outsider Unforgeability) *The proposed scheme is existential unforgeable against adaptive chosen message attack under the random oracle model assuming the hardness of the Computational Diffie-Hellman problem in G_1 .*

Proof: We consider an adversary A playing the following forgery game. A is given with the following resources:

- *Public key:* initially, the adversary is given the public key tuple PK ;
- *Hash queries:* at anytime, the adversary can ask a hash value for any input. For each hash function, we will maintain a hash list so that the hash outputs will be consistent;
- *Sign query:* at anytime, the adversary can ask for a signature on any message corresponding to any valid keystone-fix (i.e. keystone-fix that has been output by the `KGEN` hash query). The adversary will be answered with a signature that passes the `OutsiderVerify` algorithm.

The goal of the adversary is

- to construct a signature with a keystone-fix that passes the `OutsiderVerify` algorithm,
- the forgery signature is not equal to any signature that has been given to the adversary from the sign query.

Suppose that there exists an adversary that plays with non-negligible success in the above attack game, we will show that by using such adversary, we can solve the computational Diffie-Hellman problem in G_1 . That is given P , xP and yP , we can calculate xyP with non-negligible probability. We play as follows.

- Choose a random index $j \in [1, n]$. For each $i \neq j$, $i \in [0, n]$, choose a random $sk_i \in \mathbf{Z}_q$ and set $P_i = sk_i P$. Choose random $s \in \mathbf{Z}_q$ and set $P_j = s x P$. Give the adversary the public key tuple $PK = (P, P_0, P_1, P_2, \dots, P_n)$.
- For each new hash query to $\text{KGEN} = H_1$, choose a random number $t \in \mathbf{Z}_q$ and output the hash value $t y P$. If the adversary queries with an old input then just output the value recorded in the hash list. Store the t values to a T-list.
- For each sign query, use a secret key sk_i where $i \neq j$ to sign the message and give back the adversary the signature. This signature will pass the **OutsiderVerify** algorithm because it was generated by a legitimate secret key sk_i .

Suppose that the adversary can produce a forgery signature with a keystone-fix $(m, V, U_1, \dots, U_n, F)$. The requirement to pass the **OutsiderVerify** algorithm is that

$$\hat{e}(P, V) = \hat{e}(F, \sum_{i=1}^n (U_i + h_i P_i)), \quad (1)$$

where $h_i = H_3(m || i || PK || F || U_i)$.

Since the output of H_3 is totally random, the adversary must have queried H_3 with $m || i || PK || F || U_i$ for each $i = 1, \dots, n$. The probability that among these n hash queries, the query on $m || j || PK || F || U_j$ was made last is non-negligible. Thus, the hash query $H_3(m || j || PK || F || U_j) = h_j$ was made when all of the following values are known to (or chosen by) the adversary:

$$m, F, U_1, \dots, U_n, h_1, \dots, h_{j-1}, h_{j+1}, \dots, h_n.$$

From (1) we have

$$\hat{e}(P, V) = \left[\hat{e}(F, \sum_{i \neq j} (U_i + h_i P_i)) \hat{e}(F, U_j) \right] \hat{e}(F, h_j P_j).$$

Note that in the above equation, the value inside the square brackets and F and P_j are all known to the adversary at the hash query $H_3(m || j || PK || F || U_j) = h_j$. Thus, upon receiving the value h_j returned from the hash oracle, the adversary can determine with non-negligible the value of V such that

$$\hat{e}(P, V) = \left[\hat{e}(F, \sum_{i \neq j} (U_i + h_i P_i)) \hat{e}(F, U_j) \right] \hat{e}(F, h_j P_j).$$

By forking lemma, we can replay the adversary to produce another forgery $(m, V', U_1, \dots, U_n, F)$ where $H_3(m || j || PK || F || U_j) = h'_j$. Thus, we obtain another equality

$$\hat{e}(P, V') = \left[\hat{e}\left(F, \sum_{i \neq j} (U_i + h_i P_i)\right) \hat{e}(F, U_j) \right] \hat{e}(F, h'_j P_j).$$

Hence,

$$\hat{e}(P, V' - V) = \hat{e}(F, (h'_j - h_j) P_j).$$

Recall that $P_j = s x P$ and $F = t y P$ for some random t in the T-list. Thus, the above equality gives

$$\hat{e}(P, V' - V) = \hat{e}(t y P, (h'_j - h_j) s x P).$$

It follows that $V' - V = (h'_j - h_j) s t x y P$. Therefore, we can calculate the value

$$x y P = [(h'_j - h_j) s t]^{-1} (V' - V).$$

This completes the proof. ■

Inside Attacker: for any two users u and v , any group of other users cannot frame user v by producing a signature that passes the algorithm `InsiderVerify` and makes user u believe that the signature is created by user v .

Theorem 3. (Insider Unforgeability) *The proposed scheme is existential unforgeable against adaptive chosen message attack under the random oracle model assuming the hardness of the Computational bilinear Diffie-Hellman problem in (G_1, G_2, \hat{e}) .*

Proof: We consider an adversary A playing the following forgery game. A first chooses two user identities u and v . Let $X_{uv} = \{1, \dots, n\} \setminus \{u, v\}$. A is given with the following resources:

- *Public key and secret key:* initially, the adversary is given the public key tuple PK and $n - 2$ secret keys $\{sk_i : i \in X_{u,v}\}$;
- *Hash queries:* at anytime, the adversary can ask a hash value for any input. For each hash function, we will maintain a hash list so that the hash outputs will be consistent;
- *Sign query:* since the adversary has the secret key of every user i in X_{uv} , the adversary can sign any message on behalf of user $i \in X_{uv}$. In addition, we allow that, at anytime, the adversary can ask for a signature signed on the secret key sk_u or sk_v on any message corresponding to any valid keystone-fix (i.e. keystone-fix that has been output by the `KGEN` hash query). The adversary will be answered with a signature that passes the algorithm `InsiderVerify` which is executed against any secret key sk_j of user $j \in X_{uv}$.

The goal of the adversary is

- to construct a forgery signature that makes user u believe that it is created by user v , i.e. this signature must pass the `InsiderVerify` algorithm when executed with secret key sk_u against the sender v ,

- the forgery signature is not equal to any signature that has been given to the adversary from the sign query.

Suppose that there exists an adversary that plays with non-negligible success in the above attack game, we will show that by using such adversary, we can solve the computational bilinear Diffie-Hellman problem in (G_1, G_2, \hat{e}) . That is given P, xP, yP and zP , we can calculate $\hat{e}(P, P)^{xyz}$ with non-negligible probability. We play as follows.

- For each $i \in X_{uv}$, choose a random $sk_i \in \mathbf{Z}_q$ and set $P_i = sk_i P$. Choose random $s_0, s_u, s_v \in \mathbf{Z}_q$ and set $P_0 = s_0 xP, P_u = s_u yP$ and $P_v = s_v zP$. This corresponds to $sk_0 = s_0 z, sk_u = s_u y$ and $sk_v = s_v z$. Note that we do not know the values of sk_0, sk_u, sk_v but we can calculate P_0, P_u and P_v . Give the adversary the public key tuple $PK = (P, P_0, P_1, P_2, \dots, P_n)$ and $n - 2$ secret keys $\{sk_j : j \in X_{uv}\}$.
- For each new hash query to $\text{KGEN} = H_1$, choose a random number $t \in \mathbf{Z}_q$ and output the hash value tP . If the adversary queries with an old input then just output the value recorded in the hash list. Store the t values to a T-list.
- For each sign query on secret key sk_u , use the normal **Sign** algorithm to sign. There are two places in algorithm **Sign** that we need to use the secret key value sk_u :

$$\hat{e}(P_0, P_i)^{sk_u} \text{ for } i \neq u \quad \text{and} \quad V = sk_u(h_u + H_4(ik_u))F.$$

However, we do not know the value sk_u , so we need to show that we can calculate these values without explicitly use sk_u . Firstly, for $i \neq u, v$, $\hat{e}(P_0, P_i)^{sk_u}$ can be calculated as $\hat{e}(P_0, P_u)^{sk_i}$ since we know P_0, P_u and sk_i . Secondly, the value $\hat{e}(P_0, P_v)^{sk_u}$ is chosen as a random element of G_2 . And finally, we can check in the T-list for the value t such that $F = tP$, so V can be calculated as $V = t(h_u + H_4(ik_u))P_u$. This signature clearly passes the algorithm **InsiderVerify** against the sender u for any secret key sk_i of user $i \in X_{uv}$. Thus, we have shown that we can answer sign requests on secret key sk_u . Sign requests on secret key sk_v can be answered similarly.

Suppose that the adversary can produce a forgery signature $(m_v, V, U_1, \dots, U_n, F)$. To convince user u that this signature is signed by user v , it must hold that

$$U_u = H_2(m_v || v || F || \hat{e}(P_0, P_v)^{sk_u}).$$

Since the output of H_2 is totally random, the adversary must have queried H_2 with the exact input $m_v || v || F || \hat{e}(P_0, P_v)^{sk_u}$. We can look up in the hash list for this input, thus, we can obtain $\hat{e}(P_0, P_v)^{sk_u}$. However,

$$\hat{e}(P_0, P_v)^{sk_u} = \hat{e}(P, P)^{s_0 sk_u sk_v} = \hat{e}(P, P)^{s_0 s_u s_v xyz}.$$

Therefore, we can calculate the value

$$\hat{e}(P, P)^{xyz} = [\hat{e}(P_0, P_v)^{sk_u}]^{\frac{1}{s_0 s_u s_v}}.$$

This completes the proof. ■

Ambiguity. An adversary cannot prove that user j is the actual signer of a signature before the individual keystone ik_j is released, except that from the algorithm `OutsiderVerify`, the adversary knows that the actual signer is among the users.

Theorem 4. *The proposed scheme satisfies the ambiguity property.*

Proof: The difference between the algorithm `OutsiderVerify` and the algorithm `BindingVerify` is the checking that links a signature to a user j by the validity of the equation

$$\hat{e}(P, V) \stackrel{?}{=} \hat{e}(P_j, (h_j + H_4(ik_j))F).$$

This is equivalent to the relation $V = sk_j(h_j + H_4(ik_j))F$. Since the output of H_4 is totally random, before the release of the individual keystone ik_j , the value $sk_j(h_j + H_4(ik_j))F$ is totally random as a random element of G_1 , thus, from the value V in the signature, there is no way to establish the relation $V = sk_j(h_j + H_4(ik_j))F$. This completes the proof. ■

Fairness. Clearly, our scheme satisfies the fairness property because after all the keystones are released no matching signer left in a position where a keystone binds his signature to him whilst some of the other signers' signatures are not bound to them. Thus, we have the following theorem.

Theorem 5. *Our proposed multi-party concurrent signature scheme satisfies all four properties: correctness, unforgeability, ambiguity and fairness, assuming the computational bilinear Diffie–Hellman assumption in random oracle model.*

4 Conclusion

In this paper, we have constructed for the first time a multi-party concurrent signature scheme based on Chow-Yiu-Hui's ring signature scheme. There are two types of ring signatures. One with linear signature size and one with constant signature size. The Chow-Yiu-Hui's ring signature scheme belongs to the first type, thus our multi-party concurrent signature scheme has signature size $\mathcal{O}(n^2)$. The remaining as an open problem is to construct a multi-party concurrent signature scheme based on constant-size ring signatures.

References

1. M. Abe, M. Ohkubo and K. Suzuki, 1-out-of-n signatures from a variety of keys, ASIACRYPT'02, LNCS 2501, 415–432, 2002.
2. N. Asokan, V. Shoup and M. Waidner, Optimistic fair exchange of signature, EUROCRYPT'98, LNCS 1403, 591–606, 1998.
3. N. Asokan, V. Shoup and M. Waidner, Optimistic fair exchange of signature, *IEEE Journal on Selected Areas in Communication* **18** (4) (2000), 593–610.
4. D. Boneh and M. Naor, Timed commitments, CRYPTO'00, LNCS 1880, 236–254, 2000.

5. B. Baum-Waidner and M. Waidner, Round-optimal and abuse free optimistic multi-party contract signing, ICALP'00, LNCS 1853, 524–535, 2000.
6. L. Chen, C. Kudla and K.G. Paterson, Concurrent signatures, EUROCRYPT'04, LNCS 3027, 287–305, 2004.
7. S.S.M. Chow and W.Susilo, Generic construction of (identity-based) perfect concurrent signatures, ICICS'05, LNCS 3783, 194–206, 2005.
8. S.S.M. Chow, S.M. Yiu and L.C.K. Hui, Efficient identity based ring signature, ACNS'05, LNCS 3531, 499–512, 2005.
9. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup, Anonymous identification in Ad Hoc Groups, EUROCRYPT'04, LNCS 3027, 609–626, 2004.
10. Y. Dodis and L. Reyzin, Breaking and repairing optimistic fair exchange from PODC 2003, DRM'03, 47–54, 2003.
11. S. Even, O. Goldreich and A. Lempel, A randomized protocol for signing contracts, *Communications of the ACM* **28** (6), 637–647, 1985.
12. O. Goldreich, A simple protocol for signing contracts, CRYPTO'83, 133–136, 1983.
13. J. Garay, M. Jakobsson and P. MacKenzie, Abuse-free optimistic contract signing, CRYPTO'99, LNCS 1666, 449–466, 1999.
14. J. Garay and C. Pomerance, Timed fair exchange of standard signatures, FC'03, LNCS 2742, 190–207, 2003.
15. M. Jakobsson, K. Sako and R. Impagliazzo, Designated verifier proofs and their applications, EUROCRYPT'96, LNCS 1070, 143–154, 1996.
16. L. Nguyen, Accumulators from bilinear pairings and applications, CT-RSA'05, LNCS 3376, 275–292, 2005.
17. J.M. Park, E.K.P. Chong and H.J. Siegel, Constructing fair exchange protocols for e-commerce via distributed computation of RSA signatures, PODC'03, 172–181, 2003.
18. R.L. Rivest, A. Shamir and Y. Tauman, How to leak a secret, ASIACRYPT'01, LNCS 2248, 552–565, 2001.
19. W. Susilo and Y. Mu, Tripartite concurrent signatures, IFIP/SEC'05, 425–441, 2005.
20. W. Susilo, Y. Mu and F. Zhang, Perfect concurrent signature schemes, ICICS'04, LNCS 3269, 14–26, 2004.

Formal Security Model of Multisignatures

Yuichi Komano¹, Kazuo Ohta², Atsushi Shimbo¹, and Shinichi Kawamura¹

¹ Toshiba Corporation,

1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
{yuichi1.komano, atsushi.shimbo, shinichi2.kawamura}@toshiba.co.jp

² The University of Electro-Communications,

Chofugaoka 1-5-1, Chofu-shi, Tokyo 182-8585, Japan
ota@ice.uec.ac.jp

Abstract. A multisignature scheme enables multiple signers to cooperate to generate one signature for some message. The aim of the multisignatures is to decrease the total length of the signature and/or the signing (verification) costs. This paper first discusses a formal security model of multisignatures following that of the group signatures [1,4]. This model allows an attacker against multisignatures to access five oracles adaptively. With this model, we can ensure more general security result than that with the existence model [14,11,12]. Second, we propose a multisignature scheme using a claw-free permutation. The proposed scheme can decrease the signature length compared to those of existence multisignature schemes using a trapdoor one-way permutation (TWOP) [11,12], because its signing does not require the random string. We also prove that the proposed scheme is tightly secure with the formal security model, in the random oracle model. Third, we discuss the security of the multisignature schemes [11,12] using a TOWP with the formal security model to confirm that these schemes can be proven to be tightly secure.

Keywords: multisignature scheme, formal security model, claw-free permutation, random oracle model.

1 Introduction

The notion of multisignatures was introduced by Itakura and Nakamura [8], and a great deal of research has been done on this subject [18,16,17,14,15,5,10,11,12]. In a multisignature scheme, two or more signers cooperate to generate one (multi) signature for some message: The same result is accomplished by concatenating signatures generated by the signers individually; however, the aim of the multisignature scheme is to decrease the total length of the signature and/or the signing (verification) costs.

As for the provably secure multisignature schemes using a trapdoor one-way permutation (TOWP) like an RSA permutation [19], Mitomi and Miyaji¹ [15] and Kawauchi and Tada [10] proposed multisignature schemes based on the full domain hash (FDH, [2]) and the probabilistic signature scheme (PSS, [3]),

¹ See Appendix A in [15].

respectively. In these schemes, the signing is performed by signers sequentially, and the length of signature increases as often as signers proceed the signing. Since the signing is proceeded by substituting an intermediate multisignature² for a TOWP, the key length (input size of the permutation) should be longer than the signature length. Namely, if each signer publishes own key pair, the signing order is restricted by the key length. Moreover, as the signing order proceeds, the computation cost is enlarged because of the increase of the key length.

Recently, several provably secure multisignature schemes using a TOWP [11,12] in which the key length is independent from the signing order were proposed. These schemes are based on the probabilistic full domain hash (PFDH, [7]) and PSS, respectively, and are tightly secure in the random oracle model. The optimal lengths of random strings in these schemes is estimated by $\log_2 q_{\Sigma}$ bits (≈ 30 bits) [12]. Since the random string is required in the verification process, it should be attached with the signature, and hence, the length of the signature increases by $\log_2 q_{\Sigma}$ bits per a signer. Note that if a system with the multisignature scheme is desired to work for a long term (*i.e.*, each signer will generate many signatures), then the length of random string should be enlarged (in *log* order of the number of signatures). This also enlarges the signature length.

1.1 Our Contribution

This paper first formalizes a security model of multisignatures following that of the group signatures [1,4]. In our model, an attacker against multisignatures can adaptively access five oracles; add user, register, corrupt, signing, and hash oracles. Note that the previous security consideration has not discussed the registration of signers sufficiently. Our model casts light on the security in active signer registration. With this model, we can ensure more general result than that with the existence model [14,11,12].

Second, this paper proposes a multisignature scheme using a claw-free permutation (CFP), and proves that the proposed scheme is tightly secure in EUF-ACMA&AIA in the random oracle model [2], where EUF-ACMA&AIA denotes *unforgeability against adaptive chosen message attack and adaptive insider attack*. In the proposed scheme, the increase of signature length is by one bit per a signer; while, in the existence multisignature schemes using a TOWP [11,12], the increase is by 30 bits ($\approx \log_2 q_{\Sigma}$ bits) per a signer.

Third, we discuss the security of multisignature schemes [11,12] using a TOWP with the formal security model. With this model, we can show that these schemes can be also proven to be tightly secure and that the optimal length of random string can be also estimated by $\log_2 q_{\Sigma}$ bits (≈ 30 bits). The proof is fairly complicated; however, if we modify the schemes by adding the public keys of signers to input of hash function, we can easily prove their security in the same manner as the security proof of multisignature scheme using a CFP.

² More precisely, the exclusive-or of the intermediate multisignature and hash value are utilized.

1.2 Related Work

Boneh et al. [6] proposed a notion of an aggregate signatures which resembles the multisignatures. In an aggregate signature scheme, each signer individually generates her signature and an aggregator (maybe he is not one of the signers) aggregates the signatures into one signature. Note that the messages should be distinct in order to proceed the aggregation correctly.

The original concept of the multisignatures is that signers cooperate to generate one signature for the same message. Note that only the signers (not an aggregator who has no signing key) can generate the multisignature. On the other hand, Mitomi and Miyaji [15] introduced a notion of *message flexibility* to the multisignatures. The message flexibility allows signers to sign distinct messages.

The difference between the aggregate signatures and multisignatures can be explained not by the message flexibility but by the entity who joints signatures into one. From this viewpoint, the sequential aggregate signature scheme [13] should be regarded as a multisignature scheme.

2 Definitions

We first introduce the definitions of a multisignature scheme and its security, a trapdoor one-way permutation (TOWP), and a claw-free permutation (CFP).

Hereafter, we assume that the total group of signers \mathcal{G} consists of (constant ordered) L players (signers), P_1, P_2, \dots, P_L who have a pair of identification number and public key, $(ID_1, \text{pk}_1), (ID_2, \text{pk}_2), \dots, (ID_L, \text{pk}_L)$, respectively, and also assume that L' signers $P_{i_1}, P_{i_2}, \dots, P_{i_{L'}}$ in $\mathcal{G}' (\subseteq \mathcal{G})$ execute a multisigning algorithm. Note that these L' signers may be selected adaptively in the process of the multisigning algorithm³.

Definition 1 (multisignature scheme). *A multisignature scheme consists of the following three algorithms, $(\mathcal{K}, \mathcal{MS}, \mathcal{V})$.*

- *Key generation algorithm \mathcal{K} is a probabilistic algorithm which, given a security parameter k , outputs a key pair (public and private keys), $\mathcal{K}(1^k) = (\text{pk}_{i_j}, \text{sk}_{i_j})$ for each signer P_{i_j} of \mathcal{G} where $j \in [1, L]$ and $i_j \in [1, L]$.*
- *Multisigning algorithm \mathcal{MS} is performed by $P_{i_j} \in \mathcal{G}'$ where $j \in [1, L']$ and $i_j \in [1, L]$. The inputs of this algorithm are message m_{i_j} (concatenated with the previous signer's message $m_{i_{j-1}}$), a signature of $P_{i_{j-1}}$ $\sigma_{i_{j-1}}$ and secret key sk_{i_j} , and the output is a signature $\sigma_{i_j} = \mathcal{MS}_{\text{sk}_{i_j}}(m_{i_j}, \sigma_{i_{j-1}})$. This algorithm may be probabilistic.*
- *Verification algorithm \mathcal{V} takes message $m_{i_{L'}}$, multisignature $\sigma_{i_{L'}}$, and public keys of signers $\text{pk}_{i_1}, \dots, \text{pk}_{i_{L'}}$, and returns $\mathcal{V}_{\text{pk}_{i_1}, \dots, \text{pk}_{i_{L'}}}(m_{i_{L'}}, \sigma_{i_{L'}}) = 1$ if $\sigma_{i_{L'}}$ is a valid multisignature of $m_{i_{L'}}$ and \mathcal{G}' , and otherwise, returns 0 (\perp). This algorithm is deterministic.*

³ This paper discusses an order flexible multisignature scheme [15], in which each signer can decide whether she joins the signature or not in signing process.

Add(i) : If $P_i \in \text{List}$ then return \perp $(\text{pk}_i, \text{sk}_i) \leftarrow \mathcal{K}(1^k)$ $P_i \rightarrow \text{List}$ return pk_i	Crpt(i) : If $P_i \notin \text{List} \setminus \text{C-List}$ then return \perp $P_i \rightarrow \text{C-List}$ return sk_i
Reg(i, pk_i) : If $P_i \in \text{List}$ then return \perp $P_i \rightarrow \text{List}; P_i \rightarrow \text{C-List}$ return 1	$\Sigma(\text{ID}_{i_j}, m, \sigma_{i_{j-1}})$: If $P_{i_j} \notin \text{List} \setminus \text{C-List}$ then return \perp If $\sigma_{i_{j-1}}$ is invalid then return \perp return $\mathcal{MS}(\text{sk}_{i_j}, m)$
	$H(m)$: return $H(m)$

Fig. 1. Oracles

We then introduce the formal security model of multisignature schemes, following that of group signatures [1,4].

Definition 2 (formal security model of multisignatures). Let $(\mathcal{K}, \mathcal{MS}, \mathcal{V})$ be a multisignature scheme and let \mathcal{F} be an attacker against the scheme. \mathcal{F} accesses the following five oracles (Fig. 1) to mount an attack. Here, let **List** and **C-List** be lists which include the whole signer (an identity and her public key) registered into the system, and the malicious signers who are registered (with **Reg** oracle) or corrupted (with **Crpt** oracle) by \mathcal{F} , respectively.

— **Add(i)** : An *add user oracle* is invoked to add an honest signer with identity i to **List**. If a signer with identity i already exists, then the oracle returns \perp . Otherwise, the oracle runs the key generation algorithm and adds the signer (and her public key) to **List**. Finally, the oracle returns pk_i .

— **Reg(i, pk_i)** : With a *signer register oracle*, \mathcal{F} can register a new signer with public key pk_i in **List**. The oracle also adds the signer to **C-List**.

— **Crpt(i)** : A *corrupt oracle* is utilized to corrupt the signer whose identity is i . \mathcal{F} can draw the secret key sk_i of signer P_i from the oracle.

— **$\Sigma(\text{ID}_{i_j}, \text{M}_{i_j}, \sigma_{i_{j-1}})$** : A *signing oracle* is given the identity of an honest signer P_{i_j} , message M_{i_j} , and (multi)signature $\sigma_{i_{j-1}}$ to output a multisignature σ_{i_j} .

— **$H(m)$** : A *hash oracle (random oracle, [2])* is given a message m as an input and outputs a string $H(m)$.

We define the success probability of \mathcal{F} with

$$\Pr \left[\mathcal{V}_{\text{pk}_{i_1}, \text{pk}_{i_2}, \dots, \text{pk}_{i_{L'}}}(m, \sigma_{i_{L'}}) = 1 \wedge \exists j \in \{1, \dots, L'\} \text{ where } P_{i_j} \text{ is not corrupted nor asked to sign } m \text{ for } \mathcal{G}' \right].$$

Here, $\mathcal{G}' = \{P_{i_1}, P_{i_2}, \dots, P_{i_{L'}}\}$ is a subgroup of $\mathcal{G} = \{P_1, P_2, \dots, P_L\}$ chosen by \mathcal{F} adaptively.

We say that the multisignature scheme $(\mathcal{K}, \mathcal{MS}, \mathcal{V})$ is *EUF-ACMA&AIA (existential unforgeable against adaptive chosen message attack and adaptive insider attack)* secure in $(L, \tau, q_\Sigma, q_H, \epsilon)$ if there is no attacker \mathcal{F} who can achieve the success probability more than ϵ within time bound τ even if \mathcal{F} accesses the add user and signer register oracles L times in total, the corrupt, signing, and hash oracles at most $L - 1$, q_Σ , and q_H times, respectively.

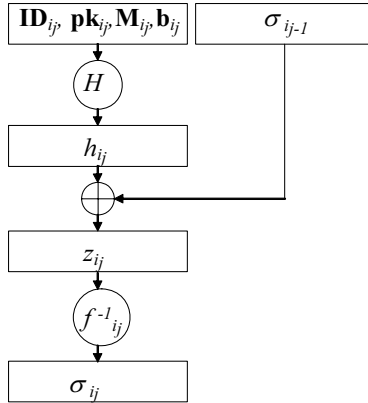


Fig. 2. Multisignature Scheme using Claw-free Permutation

We then review the definitions of the trapdoor one-way permutation (TOWP) and claw-free permutation (CFP).

Definition 3 (TOWP). Assume that $f : X \rightarrow Y$ be a bijective map. We say that a polynomial time algorithm \mathcal{A} breaks the one-wayness of f in (τ', ϵ') if \mathcal{A} , given $y \in Y$, outputs $x \in X$ such that $f(x) = y$ holds within time bound τ' and with success probability more than ϵ' . We say that f is a (τ', ϵ') trapdoor one-way permutation (TOWP) if (i) a polynomial time attacker can break the one-wayness of f with secret information (trapdoor), and (ii) no attacker can break the one-wayness of f in (τ', ϵ') without the secret information.

Definition 4 (CFP). Assume that $f, g : X \rightarrow X$ be TOWPs over the set X . We say that a polynomial time algorithm \mathcal{A} finds a claw of (f, g) in (τ', ϵ') if \mathcal{A} outputs the claw (x_1, x_2) such that $f(x_1) = g(x_2)$ holds within time bound τ' and with success probability more than ϵ' . We say that (f, g) is a pair of claw-free permutations (CFPs) if no attacker can find a claw of (f, g) in (τ', ϵ') .

3 Multisignature Scheme Using Claw-Free Permutation

In this section, we propose a multisignature scheme using a CFP and discuss its security. The proposed scheme can be regarded as a multisignature scheme based on the signature scheme using a CFP proposed by Katz and Wang [9].

3.1 Construction

Suppose that the system contains L signers P_1, P_2, \dots , and P_L with identity numbers ID_1, ID_2, \dots , and ID_L , respectively, and that the number of signers who join the multisignature by L' . For simplicity, let us assume that each signer signs only once in the signing process of one multisignature, namely assume $1 \leq L' \leq L$.

Let PKI be a trusted third party who manages public keys of signers registered in the multisignature system. When each signer P_i generates a key pair of public and private keys, P_i asks the PKI to register the public key with her identity number ID_i .

Protocol 1 (multisignature scheme using a CFP, Fig. 2). For a security parameter k , let $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ be a hash function (random oracle).

- **Key Generation:** Each signer P_i runs the key generation algorithm \mathcal{K} to generate a pair of CFPs and their inverses $(f_i, g_i, f_i^{-1}, g_i^{-1})$. Note that f_i and g_i are the permutations over $\{0, 1\}^k$. Moreover, P_i constructs a one-bit output function $S_i : \{0, 1\}^* \rightarrow \{0, 1\}$ secretly. P_i then chooses one of the CFPs f_i and g_i (without loss of generality, we assume that P_i selects f_i), publishes f_i as a public key to the PKI, and holds (f_i^{-1}, S_i) secretly as a private key.
- **Multisigning:** Signer P_{i_j} receives a previous signer's (multi)signature $(\mathbf{ID}_{i_{j-1}}, \mathbf{pk}_{i_{j-1}}, \mathbf{M}_{i_{j-1}}, \mathbf{b}_{i_{j-1}}, \sigma_{i_{j-1}})$ and, if necessary, checks its validity with the verification algorithm below. If the signer P_{i_j} is the first one (*i.e.*, $j = 0$), she regards the previous signer's signature $\mathbf{ID}_{i_0} = \mathbf{ID}_{i_0} = \phi, \mathbf{pk}_{i_0} = f_{i_0} = \phi, \mathbf{M}_{i_0} = M_{i_0} = \phi, \mathbf{b}_{i_0} = b_{i_0} = \phi, \sigma_{i_0} = 0^k$. P_{i_j} sets $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}} || \mathbf{ID}_{i_j}, \mathbf{pk}_{i_j} = \mathbf{pk}_{i_{j-1}} || f_{i_j}$, and $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}} || M_{i_j}$, computes $b_{i_j} = S_{i_j}(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_{j-1}})$, and fixes $\mathbf{b}_{i_j} = \mathbf{b}_{i_{j-1}} || b_{i_j}$. P_{i_j} finally computes $h_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j})$, and $z_{i_j} = h_{i_j} \oplus \sigma_{i_{j-1}}$; and outputs $\sigma_{i_j} = f_{i_j}^{-1}(z_{i_j})$ as a multisignature with $\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j}$.
- **Verification:** Verifier V checks the validity of a pair of message and multisignature with some additional information $(\mathbf{ID}_{i_{L'}}, \mathbf{pk}_{i_{L'}}, \mathbf{M}_{i_{L'}}, \mathbf{b}_{i_{L'}}, \sigma_{i_{L'}})$ as follows. For $j = L', L'-1, \dots, 1, \mathcal{V}$ computes the followings: For σ_{i_j} , V calculates $z_{i_j} = f_{i_j}(\sigma_{i_j})$ and $h_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j})$. V then sets $\sigma_{i_{j-1}} = h_{i_j} \oplus z_{i_j}$ and divides $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}} || \mathbf{ID}_{i_j}, \mathbf{pk}_{i_j} = \mathbf{pk}_{i_{j-1}} || f_{i_j}, \mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}} || M_{i_j}$, and $\mathbf{b}_{i_j} = \mathbf{b}_{i_{j-1}} || b_{i_j}$. When \mathcal{V} recovers σ_{i_0} for $j = 1$, \mathcal{V} checks whether $\sigma_{i_0} = 0^k$ holds or not. \mathcal{V} accepts the multisignature if the above equation holds; otherwise, rejects it.

The proposed scheme in Protocol 1 is message and order flexible, and order verifiable [15].

Note: The secret function S_i for each signer P_i in Protocol 1 can be constructed with a random oracle as follows. Suppose that H' is a random oracle. P_i chooses a secret information K_i with an arbitrary length, and defines $S_i(M)$ as the least significant bit of $H'(M, K_i)$ for a message M .

Note that no one, except the signer herself, can check the validity of the equality $b_i = S_i(M)$. Namely, the (multi)signature for message M is unique as $(M, S_i(M), f_i^{-1}(H(M, S_i(M))))$ for the signer P_i ; while, for a verifier, there are two signatures for message M , $(M, S_i(M), f_i^{-1}(H(M, S_i(M))))$ and $(M, S_i(M) \oplus 1, f_i^{-1}(H(M, S_i(M) \oplus 1)))$, acceptable in the verification algorithm.

```

if  $(i, *, *) \in \text{List}$  return  $\perp$ 
else if  $i = t$ 
     $(i, f, \phi) \rightarrow \text{List}$ 
    return  $f_t = f$ 
else
     $(f_i, f_i^{-1}, S_i) \leftarrow \mathcal{K}(1^\kappa)$ 
     $(i, f_i, f_i^{-1}) \rightarrow \text{List}$ 
    return  $f_i$ 
    
```

Fig. 3. Simulation of $\text{Add}(i)$

```

if  $(i, *, *) \in \text{List}$  return  $\perp$ 
else if  $i = t$ 
    Abort (fail the proof)
else
     $(i, f_i, \phi) \rightarrow \text{List}$ 
     $(i, f_i, \phi) \rightarrow \text{C-List}$ 
    return 1
    
```

Fig. 4. Simulation of $\text{Reg}(i, f_i)$

```

if  $(i, *, *) \notin \text{List} \setminus \text{C-List}$  then return  $\perp$ 
else if  $i = t$ 
    Abort (fail the proof)
else
     $(i, f_i, f_i^{-1}) \rightarrow \text{C-List}$ 
    return  $f_i^{-1}$ 
    
```

Fig. 5. Simulation of $\text{Crpt}(i)$

3.2 Security Consideration

We then discuss the security of the proposed scheme. The following theorem claims that the proposed scheme is EUF-ACMA&AIA secure if it is intractable to find a claw of the CFPs (f, g) .

Theorem 1. Suppose that there is an attacker who can break the proposed scheme in $(L, \tau, q_\Sigma, q_H, \epsilon)$ in the sense of EUF-ACMA&AIA. Then, we can construct an algorithm which, given a claw-free permutation (f, g) , can find a claw of (f, g) in (τ', ϵ') , where

$$\begin{cases} \epsilon' \geq \frac{1}{2L}(\epsilon - \frac{1}{2^k}) \\ \tau' \leq \tau + (q_H + q_\Sigma + 1)\mathsf{T}_f \end{cases}$$

hold. Here, T_f denotes the time complexity of f (and g) required to operate the permutation once.

Proof: Let \mathcal{F} be an attacker against the proposed scheme. The proof is done by reducing the problem of finding a claw of CFPs (f, g) to the problem of breaking the proposed scheme in EUF-ACMA&AIA, *i.e.*, constructing an algorithm \mathcal{I} which, given a security parameter k and CFPs (f, g) , uses \mathcal{F} as an oracle to output a claw (x_1, x_2) such that $f(x_1) = g(x_2)$ holds.

\mathcal{I} first selects $t \stackrel{R}{\leftarrow} [1, L]$ and regards f as the public key of signer P_t with an identity number ID_t . \mathcal{I} moreover sets a secret function $S_t : \{0, 1\}^* \rightarrow \{0, 1\}$.

\mathcal{I} then interacts with \mathcal{F} as follows. \mathcal{I} activates \mathcal{F} with the security parameter k . If \mathcal{F} outputs the add user query (Add), signer register query (Reg), corrupt query (Crpt), signing query (Σ), or hash query (H), then \mathcal{I} simulates the corresponding answer by following figures from 3 to 9, respectively. In these figures,

$\text{ID}_{i_j} = \mathbf{ID}_{i_j}, f_{i_j} = \mathbf{pk}_{i_j}, M_{i_j} = \mathbf{M}_{i_j}$ if $(i_j, *, *) \in \text{C-List}$ then return \perp else $b_{i_j} = S_{i_j}(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \phi)$ $\mathbf{b}_{i_j} = b_{i_j}$ query $\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j}$ to H to get $(0, \mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j}, *, 0^k, \sigma_{i_j}) \in \text{H-List}$ return σ_{i_j}
--

Fig. 6. Simulation of $\Sigma(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_{j-1}}, \sigma_{i_{j-1}}), |\mathbf{ID}_{i_j}| = 1$

List and C-List are the lists of a whole signers registered in the system and of the signers registered (with Reg) or corrupted (with Crpt) by \mathcal{F} , respectively. Let H-List be a list of H -queries and the corresponding answers. More precisely, it consists of octuplet $(d, \mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j}, h_{i_j}, \sigma_{i_{j-1}}, \sigma_{i_j})$. In the octuplet, d indicates whether some information for finding a claw is embedded with the query ($d = 1$) or not ($d = 0$). Note that $d = 0$ indicates the octuplet has been simulated in case \mathcal{F} queries $(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_{j-1}}, \sigma_{i_{j-1}})$ to Σ . As for the other elements, the triplet $(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j})$ is a query to H and h_{i_j} is the corresponding answer. The pair $(\sigma_{i_{j-1}}, \sigma_{i_j})$ or $(*, \tau_{i_j})$ is utilized in the simulation of the answer h_{i_j} .

\mathcal{I} finally achieves the forgery $(\mathbf{ID}_{i_{L'}}^*, \mathbf{pk}_{i_{L'}}^*, \mathbf{M}_{i_{L'}}^*, \mathbf{b}_{i_{L'}}^*, \sigma_{i_{L'}}^*)$ from \mathcal{F} . Note that the forgery passes the verification algorithm. If $\mathbf{ID}_{i_{L'}}^*$ does not contain ID_t , then \mathcal{I} aborts (fails the proof). Note that \mathcal{I} successes the guess of t with probability more than $\frac{1}{L}$.

If $\mathbf{ID}_{i_{L'}}^*$ contains ID_t , then \mathcal{I} searches the octuplet $(d, \mathbf{ID}_{i_u}^*, \mathbf{pk}_{i_u}^*, \mathbf{M}_{i_u}^*, \mathbf{b}_{i_u}^*, h'_{i_u}, *, \sigma'_{i_u}) \in \text{H-List}$ with u such that $i_u = t$. If such octuplet does not exist in H-List, then \mathcal{I} aborts (fails the proof). This failure happens with probability $\frac{1}{2^k}$ from the property of the random oracle.

If the octuplet $(d, \mathbf{ID}_{i_u}^*, \mathbf{pk}_{i_u}^*, \mathbf{M}_{i_u}^*, \mathbf{b}_{i_u}^*, h'_{i_u}, *, \sigma'_{i_u}) \in \text{H-List}$ exists, then \mathcal{I} checks whether $d = 1$ or not. If $d = 1$, then $f(\sigma_{i_u}^*) = h' = g(\sigma'_{i_u})$ holds and \mathcal{I} can find a claw $(\sigma_{i_u}^*, \sigma'_{i_u})$. Otherwise, namely if $d = 0$, since $\sigma_{i_u}^* = \sigma'_{i_u}$, \mathcal{I} cannot find a claw and fails the proof. Since S_t is a secret function of P_t (simulated by \mathcal{I}), in \mathcal{F} 's view, $S_{i_u}(\mathbf{ID}_{i_u}^*, \mathbf{pk}_{i_u}^*, \mathbf{M}_{i_u}^*, \mathbf{b}_{i_{u-1}}^*)$ is uniformly distributed in $\{0, 1\}$. Therefore, the probability with which \mathcal{I} fails because of $b_{i_u}^* = S_{i_u}(\mathbf{ID}_{i_u}^*, \mathbf{pk}_{i_u}^*, \mathbf{M}_{i_u}^*, \mathbf{b}_{i_{u-1}}^*)$ is $\frac{1}{2}$.

Therefore, the success probability ϵ' and running time τ' of \mathcal{I} are estimated by $\epsilon' \geq \frac{1}{2L}(\epsilon - \frac{1}{2^k})$ and $\tau' \leq \tau + (q_H + q_\Sigma + 1)\mathsf{T}_f$. \blacksquare

4 Reconsideration of Multisignatures Using TOWP

References [11,12] proposed multisignature schemes using a TOWP and discussed their security with an *ad-hoc* security model. This section reconsiders the security of multisignature schemes [11,12] with the formal security model.

We can show that the original schemes [11,12] can be proven to be tightly secure with the formal security model, too, and that the optimal length for the

```

IDij-1 || IDij = IDij, pkij-1 || fij = pkij, Mij-1 || Mij = Mij
if ij = t
    bij = Sij(IDij, pkij, Mij, bij-1)
    bij = bij-1 || bij
    query IDij, pkij, Mij, bij to H
        to get (0, IDij, pkij, Mij, bij, hij, σ'ij-1, σij)
        if σ'ij-1 = σ'ij then return σij
        else then return “σij-1 is invalid.”
else
    bij = Sij(IDij, pkij, Mij, bij-1)
    bij = bij-1 || bij
    query IDij, pkij, Mij, bij to H
        to get (d, IDij, pkij, Mij, bij, hij, σ'ij-1, σij)
        if d = 0
            if σ'ij-1 = σ'ij then return σij
            else then return “σij-1 is invalid.”
        else
            for u = j down to v such that iv-1 = t
                σiu-1 = fiu(σiu) ⊕ hiu
            if fi(σt) = g(τt)
                output σt, τt as a claw of (f, g)
            else
                return ⊥

```

Fig. 7. Simulation of $\Sigma(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_{j-1}}, \sigma_{i_{j-1}}), |\mathbf{ID}_{i_j}| > 1$

```

IDij = IDij, fij = pkij, Mij = Mij, bij = bij
if ij = t
    if bij = Sij(IDij, pkij, Mij, ϕ)
        σij ←R {0, 1}k
        hij = fij(σij)
        (0, IDij, pkij, Mij, bij, hij, 0k, σij) → H-List
        return hij
    else
        τij ← {0, 1}k
        hij = g(τij)
        (1, IDij, pkij, Mij, bij, hij, ϕ, τij) → H-List
        return hij
else
    σij ←R {0, 1}k
    hij = fij(σij)
    (0, IDij, pkij, Mij, bij, hij, 0k, σij) → H-List
    return hij

```

Fig. 8. Simulation of $H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j}), |\mathbf{ID}_{i_j}| = 1$

```

IDi-1 || IDi = IDi, pki-1 || fi = pki, Mi-1 || Mi = Mi, bi-1 || bi
if ij = t
  if bi = Si(IDi, pki, Mi, bi-1)
    query IDi-1, pki-1, Mi-1, bi-1 to H
      to get (0, IDi-1, pki-1, Mi-1, bi-1, hi-1, *, σi-1)
      σi ←R {0, 1}k
      hi = fi(σi) ⊕ σi-1
      (0, IDi, pki, Mi, bi, hi, σi-1, σi) → H-List
      return hi
  else
    query IDi-1, pki-1, Mi-1, bi-1 to H
      to get (0, IDi-1, pki-1, Mi-1, bi-1, hi-1, *, σi-1)
      τi ← {0, 1}k
      hi = g(τi) ⊕ σi-1
      (1, IDi, pki, Mi, bi, hi, φ, τi) → H-List
      return hi
else
  query IDi-1, pki-1, Mi-1, bi-1 to H
    to get (d, IDi-1, pki-1, Mi-1, bi-1, hi-1, *, σi-1)
  if d = 0
    σi ←R {0, 1}k
    hi = fi(σi) ⊕ σi-1
    (0, IDi, pki, Mi, bi, hi, σi-1, σi) → H-List
    return hi
  else
    hi ←R {0, 1}k
    (1, IDi, pki, Mi, bi, hi, *, *) → H-List
    return hi

```

Fig. 9. Simulation of $H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{b}_{i_j})$, $|\mathbf{ID}_{i_j}| > 1$

length of random string can be also estimated by $\log_2 q_\Sigma$ bits (≈ 30 bits). The proof, however, is a fairly complicated and tedious one; therefore, we omit it.

In order to achieve multisignature schemes which are tightly secure with the formal model (whose security proofs are easier than those for the original schemes), we slightly modify the multisignature schemes [11,12]. In the multisignatures [11,12], the input of hash function includes identification numbers \mathbf{ID}_{i_j} in the same manner. In order to ensure the provable security with the formal security model, we add public keys \mathbf{pk}_{i_j} to the input of hash function, as the multisignature scheme using a CFP described in section 3. See Appendix for their construction and security results.

5 Discussion

References [11,12] proposed multisignature schemes using a TOWP. These schemes utilize a random string in order to ensure the tight security, and the

optimal length for the string is $\log_2 q_\Sigma$ bits (≈ 30 bits) [12]. Since the random string is required in the verification process, it should be attached with the resulting signature, and hence, the signature length⁴ increases by $\log_2 q_\Sigma$ bits (≈ 30 bits) per a signer.

On the other hand, the increase of signature length in the proposed scheme (using a CFP) is one bit per a signer. The proposed scheme can decrease the increase by $\log_2 q_\Sigma - 1$ bits (≈ 29 bits) per a signer, and it is more effective if the number of signers is increased. It is important that, in the proposed scheme, the increase does not depend on the number of signatures (q_Σ) published in the system. Note also that the proposed scheme relies its security on the stronger assumption (claw-freeness) than the one-wayness. These imply that there is a trade-off between the strength of assumption and the signature length.

With regard to the security, the proposed scheme is sufficiently tight security ($\epsilon' \approx \frac{1}{2L}\epsilon$) compared to the multisignature scheme [15] based on the full domain hash [2].

6 Conclusion

This paper first discussed a formal security model of multisignatures. With the model, in which the attacker can access five oracles adaptively, we can ensure more general result than that with the existence model [14,11,12].

Second, this paper constructed a multisignature scheme using a claw-free permutation (CFP) and proved its security based on the hardness of breaking the claw-freeness in the random oracle model. The proposed scheme uses the secret function for each signer which maps $\{0,1\}^*$ to $\{0,1\}$; while the scheme does not require the random string. It may not be adequate to compare the proposed scheme with the previous multisignature schemes [11,12] because these schemes do not use the claw-free permutation but a trapdoor one-way permutation (TOWP); however, the proposed scheme can decrease the increase of signature length compared to them.

Third, we discuss the security of the multisignature schemes [11,12] using a TOWP with the formal security model to confirm that these schemes can be proven to be tightly secure.

References

1. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

⁴ The signature length is sum of the length of message, signature, and additional information (identification number of signer, random string, etc.).

3. M. Bellare and P. Rogaway. The exact security of digital signatures –how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, Heidelberg, New York, 1996. Springer-Verlag.
4. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptography – CT-RSA 2005*, *Lecture Notes in Computer Science*, Berlin, Heidelberg, New York, 2005. Springer-Verlag.
5. A. Boldyreva. Threshold signatures, multisignatures, and blind signatures based on the gap-diffie-hellman-ground signature scheme. In Y. G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
6. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiability encrypted signature form biliner maps. In *Advances in Cryptology — EUROCRYPT 2003*, *Lecture Notes in Computer Science*, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
7. J. S. Coron. Optimal security proofs for PSS and other signature schemes. In L. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Berlin, Heidelberg, New York, 2002. Springer-Verlag.
8. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71), pages 1–8, 1983.
9. J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *CCS'03, 10th ACM Conference on Computer and Communications Security*, 2003.
10. K. Kawauchi and M. Tada. On the exact security of multisignature schemes based on RSA. In *The Eighth Australasian Conference on Information Security and Privacy (ACISP 2003)*, volume 2727 of *Lecture Notes in Computer Science*, pages 336–349, Berlin, Heidelberg, New York, 2003. Springer-Verlag.
11. K. Kawauchi and M. Tada. On the security and the efficiency of multi-signature schemes based on a trapdoor one-way permutation. *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, E88–A(5):1274–1282, 2005.
12. Y. Komano, K. Ohta, A. Shimbo, and S. Kawamura. In E. Dawson and S. Vaudenay, editors, *Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 132–150, Berlin, Heidelberg, New York, 2005. Springer-Verlag.
13. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90, Berlin, Heidelberg, New York, 2004. Springer-Verlag.
14. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *CCS'01, Eighth ACM Conference on Computer and Communications Security*, 2001.
15. S. Mitomi and A. Miyaji. A general model of multisignature schemes with message flexibility, order flexibility, and order verifiability. In *IEICE Transaction of fundamentals*, volume E-84-A, pages 2488–2499, 2001.
16. K. Ohta and T. Okamoto. A digital multisignature scheme based on the fiat-shamir scheme. In R. Rivest H. Imai and T. Matsumoto, editors, *Advances in Cryptology — Asiacrypt'91*, volume 739 of *Lecture Notes in Computer Science*, pages 139–148, Berlin, Heidelberg, New York, 1991. Springer-Verlag.

17. K. Ohta and T. Okamoto. Multi-signature schemes secure against active insider attacks. *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1):21–31, 1999.
18. T. Okamoto. A digital multisignature scheme using bijective public-key cryptosystems. *ACM Transactions on Comp. Systems*, 6(8):432–441, 1988.
19. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

A Modification to Multisignature Schemes Using TOWP

In this section, we modify two multisignature schemes using a TWOP in order to ensure the tight security with the formal security model. One of the schemes is based on the probabilistic full domain hash (PFDH, [7]), named PFDH-MSS [11], and the other is based on PSS with short signature, named S-PSS-MSS [12]. Let f_{i_j} and its inverse $f_{i_j}^{-1}$ be k bit permutations⁵. We first give the description of (modified) PFDH-MSS [11].

Protocol 2 (PFDH-MSS [11], Fig. 10 (a)). For a security parameter k , let $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ be a hash function (random oracle).

- **Key Generation:** Each signer P_i runs the key generation algorithm \mathcal{K} with a security parameter k and gets a pair of a TOWP and its inverse (f_i, f_i^{-1}) . P_i publishes f_i as a public key and holds f_i^{-1} secretly as a private key.
- **Multisigning:** Assume that signer P_{i_j} in $\mathcal{G}' = \{P_{i_1}, \dots, P_{i_{L'}}\} \subseteq \mathcal{G}$ signs on a message m_{i_j} . P_{i_j} , given identification number $\mathbf{ID}_{i_{j-1}}$, public keys $\mathbf{pk}_{i_{j-1}}$, message $\mathbf{M}_{i_{j-1}}$, random string $\mathbf{r}_{i_{j-1}}$, and signature $\sigma_{i_{j-1}}$ from a previous signer $P_{i_{j-1}}$ ($\mathbf{ID}_{i_0} = \mathbf{pk}_{i_0} = \mathbf{M}_{i_0} = \mathbf{r}_{i_0} = \phi$ and $\sigma_{i_0} = 0^k$), chooses $r_{i_j} \xleftarrow{R} \{0, 1\}^{k_0}$. Then, P_{i_j} sets $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}} || ID_{i_j}$, $\mathbf{pk}_{i_j} = \mathbf{pk}_{i_{j-1}} || f_{i_j}$, $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}} || m_{i_j}$, and $\mathbf{r}_{i_j} = \mathbf{r}_{i_{j-1}} || r_{i_j}$, and computes $\tau_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{r}_{i_j}) \oplus \sigma_{i_{j-1}}$ and $\sigma_{i_j} = f_{i_j}^{-1}(\tau_{i_j})$. Finally, P_{i_j} gives \mathbf{ID}_{i_j} , \mathbf{pk}_{i_j} , \mathbf{M}_{i_j} , \mathbf{r}_{i_j} , and σ_{i_j} to the next signer.
- **Verification:** For $\mathbf{ID}_{i_{L'}}$, $\mathbf{pk}_{i_{L'}}$, $\mathbf{M}_{i_{L'}}$, $\mathbf{r}_{i_{L'}}$, and $\sigma_{i_{L'}}$, the verifier V checks the validity of the signature by recovering σ_{i_j} ($j = L', \dots, 0$) and checking the validity of σ_{i_0} as follows: for \mathbf{ID}_{i_j} , \mathbf{pk}_{i_j} , \mathbf{M}_{i_j} , \mathbf{r}_{i_j} , and σ_{i_j} , we first compute $\tau_{i_j} = f_{i_j}(\sigma_{i_j})$. V recovers $\sigma_{i_{j-1}} = H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \mathbf{r}_{i_j}) \oplus \tau_{i_j}$. Then, V divides $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}} || ID_{i_j}$, $\mathbf{pk}_{i_j} = \mathbf{pk}_{i_{j-1}} || f_{i_j}$, $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}} || m_{i_j}$, and $\mathbf{r}_{i_j} = \mathbf{r}_{i_{j-1}} || r_{i_j}$, and executes the verification for i_{j-1} -th signer's signature. Finally, if $\sigma_{i_0} = 0^{k-1}$, V accepts the signature. Otherwise, V rejects it.

We then give the description of (modified) S-PSS-MSS [12]. Here, $|x|$ denotes the bit length of x .

Protocol 3 (S-PSS-MSS [12], Fig. 10 (b)). For a security parameter k , let $G : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k-k_0-k_1}$ and $H : \{0, 1\}^* \rightarrow \{0, 1\}^{k_1}$ be hash functions (random oracles).

⁵ For a specific f_{i_j} , like the RSA permutation [19], see the reference [12] for detail.

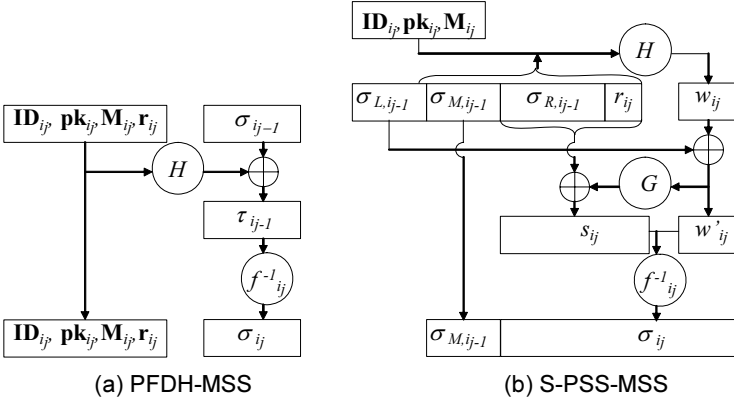


Fig. 10. Multisignature Scheme using Claw-free Permutation

- **Key Generation:** Key generation algorithm \mathcal{K} is the same as that described in the Protocol 2.
- **Multisigning:** Assume that signer P_{i_j} in $\mathcal{G}' = \{P_{i_1}, \dots, P_{i_{L'}}\} \subseteq \mathcal{G}$ signs on a message m_{i_j} . P_{i_j} , given identification number $\mathbf{ID}_{i_{j-1}}$, public keys $\mathbf{pk}_{i_{j-1}}$, message $\mathbf{M}_{i_{j-1}}$, and signature $\sigma_{i_{j-1}}$ from a previous signer $P_{i_{j-1}}$ ($\mathbf{ID}_{i_0} = \mathbf{pk}_{i_0} = \mathbf{M}_{i_0} = \phi$ and $\sigma_{i_0} = 0^{k-k_0+k_1}$), first chooses $r_{i_j} \xleftarrow{R} \{0, 1\}^{k_0}$ and divides $\sigma_{i_{j-1}}$ ($(k + (j-1)k_0)$ bits) into three parts; the least significant $k - k_0 - k_1$ bits, $\sigma_{R,i_{j-1}}$, the most significant k_1 bits, $\sigma_{L,i_{j-1}}$, and the remaining part, $\sigma_{M,i_{j-1}}$. Then, P_{i_j} sets $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}} || \mathbf{ID}_{i_j}$, $\mathbf{pk}_{i_j} = \mathbf{pk}_{i_{j-1}} || f_{i_j}$, and $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}} || m_{i_j}$, and computes $w'_{i_j} = H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j})$, $w_{i_j} = \sigma_{L,i_{j-1}} \oplus w'_{i_j}$, and $s_{i_j} = (\sigma_{R,i_{j-1}} || r_{i_j}) \oplus G(w_{i_j})$. P_{i_j} generates a signature $\sigma'_{i_j} = f_{i_j}^{-1}(s_{i_j} || w_{i_j})$. Finally, P_{i_j} gives \mathbf{ID}_{i_j} , \mathbf{pk}_{i_j} , \mathbf{M}_{i_j} , and $\sigma_{i_j} = \sigma_{M,i_{j-1}} || \sigma'_{i_j}$ to the next signer.
- **Verification:** For $\mathbf{ID}_{i_{L'}}$, $\mathbf{pk}_{i_{L'}}$, $\mathbf{M}_{i_{L'}}$, and $\sigma_{i_{L'}}$, the verifier V checks the validity of the signature by recovering σ_{i_j} ($j = L', \dots, 0$) and checking the validity of σ_{i_0} as follows: For \mathbf{ID}_{i_j} , \mathbf{pk}_{i_j} , \mathbf{M}_{i_j} , and σ_{i_j} , V first divides σ_{i_j} into two parts; lower k bits, σ'_{i_j} , and the remaining part, $\sigma_{M,i_{j-1}}$. Then, V computes $s_{i_j} || w_{i_j} = f_{i_j}(\sigma'_{i_j})$, where $|s_{i_j}| = k - k_1 - 1$ and $|w_{i_j}| = k_1$. V recovers $\sigma_{R,i_{j-1}} || r_{i_j} = s_{i_j} \oplus G(w_{i_j})$ and $\sigma_{L,i_{j-1}} = H(\mathbf{ID}_{i_j}, \mathbf{pk}_{i_j}, \mathbf{M}_{i_j}, \sigma_{M,i_{j-1}}, \sigma_{R,i_{j-1}}, r_{i_j}) \oplus w_{i_j}$, where $|\sigma_{R,i_{j-1}}| = k - k_0 - k_1$ and $|r_{i_j}| = k_0$. V then divides $\mathbf{ID}_{i_j} = \mathbf{ID}_{i_{j-1}} || \mathbf{ID}_{i_j}$, $\mathbf{pk}_{i_j} = \mathbf{pk}_{i_{j-1}} || f_{i_j}$, and $\mathbf{M}_{i_j} = \mathbf{M}_{i_{j-1}} || m_{i_j}$, and recovers $\sigma_{i_{j-1}} = \sigma_{L,i_{j-1}} || \sigma_{M,i_{j-1}} || \sigma_{R,i_{j-1}}$, and executes the verification for i_{j-1} -th signer's signature. Finally, if $\sigma_{i_0} = 0^{k-k_0+k_1}$, V accepts the signature. Otherwise, V rejects it.

As for the security, the following theorems hold. The strategy of proofs is almost the same as that of section 3.2.

Theorem 2 (Security of PFDH-MSS). Let \mathcal{F} be an attacker who breaks PFDH-MSS in $(L, \tau, q_\Sigma, q_H, \epsilon)$ in accordance with EUF-ACMA&AIA. Then we can break the one-wayness of f within time bound τ' and with success probability $\text{Succ}^{\text{ow}}(\tau')$:

$$\begin{cases} \text{Succ}^{\text{ow}}(\tau') \geq \frac{1}{L}(\epsilon - \frac{q_{\Sigma}(q_H+q_{\Sigma})}{2^{k_0}} - \frac{q_{\Sigma}+1}{2^{k-1}}) \\ \tau' \leq \tau + ((q_H + q_{\Sigma}) + L)\mathbb{T}_f \end{cases}$$

where \mathbb{T}_f denotes the time complexity of f .

Theorem 3 (Security of S-PSS-MSS). *Let \mathcal{F} be an attacker who breaks S-PSS-MSS in $(L, \tau, q_{\Sigma}, q_G, q_H, \epsilon)$ in accordance with EUF-ACMA&AIA. Then we can break the one-wayness of f within time bound τ' and with success probability $\text{Succ}^{\text{ow}}(\tau')$:*

$$\begin{cases} \text{Succ}^{\text{ow}}(\tau') \geq \frac{1}{L}(\epsilon - \frac{q_H q_{\Sigma}}{2^{k_0}} - \frac{q_H((q_H+q_{\Sigma})^2+q_G)+q_{\Sigma}(q_G+q_H+q_{\Sigma})+1}{2^{k_1}}) \\ \tau' \leq \tau + ((q_H + q_{\Sigma}) + L)\mathbb{T}_f \end{cases}$$

where \mathbb{T}_f denotes the time complexity of f .

The optimal lengths of the random string in PFDH-MSS and S-PSS-MSS are estimated by $\log_2 q_{\Sigma}$ bits, respectively [12].

Cryptanalysis of Variants of UOV

Yuh-Hua Hu¹, Chun-Yen Chou², Lih-Chung Wang³, and Feipei Lai⁴

¹ Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

d92015@csie.ntu.edu.tw

² Department of Mathematical Education, National Hualien University of Education, Hualien 970, Taiwan

choucy@mail.nhlue.edu.tw

³ Department of Applied Mathematics, National Donghwa University, Hualien 974, Taiwan

lcwang@mail.ndhu.edu.tw

⁴ Departments of Electrical Engineering & of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

flai@ntu.edu.tw

Abstract. The Unbalanced Oil and Vinegar scheme (UOV) is a signature scheme based on multivariate quadratic equations. It has o oil variables and v vinegar variables. UOV has m equations and n variables, where $m = o$ and $n = v + o$. In this paper, we define the weak key of UOV and study how to find the weak key from the public key. Second, we study the security when $m > o$. And our result shows that the security strengths of the current version of TTS, TRMS, Rainbow and MFE are $2^{59} \sim 2^{67.6}$ 3DES operations.

Keywords: multivariate quadratic, digital signature, UOV, XL.

1 Introduction

The "Oil and Vinegar" signature was proposed by Patarin [10]. The idea consists in hiding quadratic equations in o unknowns called "oil variables", v unknowns called "vinegar variables" and $v = o$. This scheme was broken by Aviad Kipnis and Adi Shamir [8]. Aviad Kipnis et al. [9] proposed a variation of the original signature called "Unbalanced Oil and Vinegar" signature. The number of vinegar variables is more than the number of oil variables, i.e. $v > o$.

The idea of the attack against "Oil and Vinegar" [8] is to separate the oil and the vinegar variables such that the attacker can extract an isomorphic copy of the private key from the public key. Kipnis et al. [9] extended the idea of [8] for UOV and its expected complexity of this attack is approximately $C_{UOV}(q, v, o) = q^{v-o-1}o^4$. Braeken et al. [2] showed that the case $v \geq o$ is particularly vulnerable to Gröbner basis attacks. Solving some quadratic equations in some variables is an intuitive way to attack MQ schemes. Yang et al. [15] stated that FXL is the best method for MQ schemes and they also proposed its complexity estimation.

In this paper, we study a variation of UOV, which the number of equations is more than o . The result shows that when the number of equations is v , the security strength is at the peak. Moreover, we review applications of the multi-layer UOV, like MFE [14], Rainbow [5], TRMS [13] and TTS [16], and show their security strengths are lower than they are claimed.

In Section 2, we describe UOV and its attacks. In Section 3, we show and analyze our method. In Section 4, we apply our method to the multi-layer UOV. And our conclusion is in Section 5.

2 UOV and Its Attacks

We introduce MQ schemes and describe UOV and its attacks.

2.1 MQ Scheme

For a typical MQ scheme, it operates on a base field \mathbb{K} . And its public key is composed of three maps, $T \circ P \circ S$, and its private key is the triple (T^{-1}, P, S^{-1}) . $S(X) = M_S X + V_S$ and $T(X) = M_T X + V_T$ are affine transformations in \mathbb{K}^n and \mathbb{K}^m respectively, where $M_S \in \mathbb{K}^{n \times n}$, $V_S \in \mathbb{K}^n$, $M_T \in \mathbb{K}^{m \times m}$, and $V_T \in \mathbb{K}^m$. S^{-1} and T^{-1} are their inverse transformations. P is a quadratic transformation and the structures of P in MQ schemes are different. It is well known that finding a solution for m general equations in n variables is a NP-complete problem [6].

2.2 UOV

Kipnis et al. [9] proposed a variation of "Oil and Vinegar" signature called "Unbalanced Oil and Vinegar". In UOV, the number of vinegar variables is more than the number of oil variables.

Private Key. The private key is the pair (P, S^{-1}) . S^{-1} is the inverse of the affine transformation, S . $P = (y_1, \dots, y_m)$ is a polynomial map with each component in the following form, where for $1 \leq i \leq m = o, 1 \leq j \leq v, 1 \leq k \leq n = v + o, a_{ijk}, b_{ik} \in \mathbb{K}$.

$$y_i = [x_1 \cdots x_n] \left(\begin{bmatrix} a_{i11} & \cdots & a_{i1v} & a_{i1(v+1)} & \cdots & a_{i1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{iv1} & \cdots & a_{ivv} & a_{iv(v+1)} & \cdots & a_{ivn} \\ a_{i(v+1)1} & \cdots & a_{i(v+1)v} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{in1} & \cdots & a_{inv} & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{i1} \\ \vdots \\ b_{in} \end{bmatrix} \right).$$

x_1, \dots, x_v are called vinegar variables and x_{v+1}, \dots, x_n are called oil variables. If a multivariate polynomial is like the above form, we call it a kernel form of UOV.

Public Key. The public key is the composed of $P \circ S = (y'_1, \dots, y'_m)$. Each component is as follows. $\alpha_{ijk}, \beta_{ij} \in \mathbb{K}$.

$$y'_i = [x'_1 \cdots x'_n] \begin{pmatrix} \alpha_{i11} & \cdots & \alpha_{i1v} & \cdots & \alpha_{i1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_{iv1} & \cdots & \alpha_{ivv} & \cdots & \alpha_{ivn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_{in1} & \cdots & \alpha_{inv} & \cdots & \alpha_{inn} \end{pmatrix} \begin{bmatrix} x'_1 \\ \vdots \\ x'_n \end{bmatrix} + \begin{bmatrix} \beta_{i1} \\ \vdots \\ \beta_{in} \end{bmatrix}.$$

Signing. To sign a message **Msg**, first we compute its hash $\mathbf{Z} = H(\mathbf{Msg}) \in \mathbb{K}^m$ by a publicly agreed hash function. Then choose v random numbers r_1, r_2, \dots, r_v . Then get x_{v+1}, \dots, x_{v+o} by solving linear equations $z_i = y_i(r_1, \dots, r_v, x_{v+1}, \dots, x_{v+o})$, where $1 \leq i \leq m$. If there is no solution, repeat to choose random numbers and solve again. Finally, the signature $\mathbf{Sig} = S^{-1}(\mathbf{X})$.

Verifying. To verify a signature **Sig**, simply check whether $V(\mathbf{Sig}) = (P \circ S)(\mathbf{Sig}) = H(\mathbf{Msg})$ holds.

2.3 Extension Attack Against Balanced Oil and Vinegar

The original attack against the Balanced Oil and Vinegar scheme is from [8]. It separates the oil and vinegar variables such that the attacker can forge signatures. We focus on the quadratic terms of the public and private keys. Let G_i , for $1 \leq i \leq m$, be the respective matrix of the quadratic form of y'_i in the public key. The quadratic part of the equations in P is represented as a quadratic form with a corresponding $(v + o) \times (v + o)$ matrix of the form: $\begin{bmatrix} A_i & B_i \\ C_i & O \end{bmatrix}$, where A_i, B_i, C_i, O are $v \times v, v \times o, o \times v, o \times o$ submatrices and O is a zero matrix. After applying $S = M_S x + V_S$, $G_i = M_S^T \begin{bmatrix} A_i & B_i \\ C_i & O \end{bmatrix} M_S$. It is clear that $\begin{bmatrix} A_i & B_i \\ C_i & O \end{bmatrix}$ maps the oil subspace ($x_1 = x_2 = \dots = x_v = 0$) to the vinegar subspace ($x_{v+1} = x_{v+2} = \dots = x_{v+o} = 0$). And if G_i is invertible, $G_i^{-1} G_j$ maps the oil space to the oil space. Therefore, the oil subspace is invariant subspace under M_S , called O_M , is a common invariant subspace of all $G_i^{-1} G_j$. And Kipnis et al. show two algorithm to find O_M , and choose a spaces V_M for $V_M + O_M = \mathbb{K}^n$ [8]. Now, we can separate the oil and vinegar variables and forge signatures.

Aviad Kipnis et al. extended the attack against the "Balanced Oil and Vinegar" to UOV scheme and gives the complexity estimate $C_{UOV}(q, v, o) = q^{v-o-1} o^4$ [9].

2.4 Solving Multivariate Quadratic Equations

XL [4] is a method to solve equations. Among all XL family, FXL seems to have the best performance [16]. XL stands for "eXtended Linearizations". XL consists the following steps:

1. Multiply: Generate all equations $x_i * y_j$. x_i is the variable in equations. y_j is the multivariate equation.
2. Linearize: Consider each monomial in all equations as a new variable and perform Gaussian elimination.
3. Solve: If the result of previous step has a univariate equation, we solve this equation and try other equations if possible.
4. Repeat: If Step 3 can not solve all variables, we treat the equations generated in Step 1 as new equations and repeat.

We can get a solution when the degree of the monomial in XL is D . There are at most $E = \binom{n+D-2}{D-2} \times m$ linear equations in $V = \binom{n+D}{D}$ variables. Hence we can consider XL as solving a lot of linear equations. There are two questions here. How to determine the minimum of D and the complexity of solving E linear equations in V variables?

Yang et al. give the rigorous estimation of D [16]. D is not easy to expressed as a function of m, n . $[x^d]f(x)$ is the coefficient of x^d in $f(x)$.

$$D = \min\{d \mid [x^d]((1-x)^{m-n-1}(1+x)^m) < 0\} \tag{1}$$

Yang et al. also list several estimations to linearize a big matrix. In solving MQ schemes, the matrix generated by quadratic equations is sparse. We chose the complexity estimation in Section 6.2 in [16].

$$C_L(E, V, q) \approx (c_0 + c_1 \lg V)EVq \tag{2}$$

q is the number of monomials in each equation and $c_0 = 4, c_1 = \frac{1}{4}$. The complexity of XL is the complexity of finding D and linearizing the big matrix generated by XL. $C_D(m, n)$ is the complexity to find D .

$$C_{XL}(m, n) = C_D(m, n) + C_L\left(\binom{n+D-2}{D-2} \times m, \binom{n+D}{D}, \frac{n(n+1)}{2}\right) \tag{3}$$

In the original UOV, m is less than n . We can fix $n - m$ variables to random values. Then the number of the variables is equivalent to the number of equations. This idea equals to add $n - m$ equations to find a solution in attacking HFE with Gröbner Basis [3]. Hence the complexity of XL to UOV is $C_{XL}(m, n) = C_{XL}(m, m)$.

3 Our Attack

3.1 Weak Keys of UOV

Definition 1. *If M in the affine transformation of the private key of UOV is like M_w in Fig. 1, then the private key is a weak key.*

Let (\hat{P}, \hat{S}) be a private key of UOV and $\hat{S} = M_{\hat{S}}x + V_{\hat{S}}$ be an affine transformation. If $M_{\hat{S}}$ is a matrix like in Fig. 1 and \hat{P} is a kernel form of UOV, $\hat{P} \circ \hat{S}$ is still a kernel form of UOV. Hence one can find another private key $(\hat{P} \circ \hat{S}, I)$, I is the identity matrix.

$$M_w = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1v} & 0 & \cdots & 0 \\ r_{21} & r_{22} & \cdots & r_{2v} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r_{v1} & r_{v2} & \cdots & r_{vv} & 0 & \cdots & 0 \\ r^{(v+1)1} & r^{(v+1)2} & \cdots & r^{(v+1)v} & r^{(v+1)(v+1)} & \cdots & r^{(v+1)(v+o)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r^{(v+o)1} & r_{n2} & \cdots & r^{(v+o)v} & r^{(v+o)(v+1)} & \cdots & r^{(v+o)(v+o)} \end{bmatrix}$$

Fig. 1. A Matrix Transformation of Weak Keys in UOV

3.2 Steps of Our Method Against UOV

Assume the public key is $\hat{P} \circ \hat{S}$ and the private key is (\hat{P}, \hat{S}) . Let $\hat{S} = M_{\hat{S}}x + V_{\hat{S}}$, $M_{\hat{S}} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, A, B, C and D are $v \times v$, $v \times o$, $o \times v$ and $o \times o$ submatrices respectively. If A is an invertible matrix, we can find a matrix $M_u = \begin{bmatrix} I_v & A^{-1}B \\ O & I_o \end{bmatrix}$, where I_v and I_o are $v \times v$ and $o \times o$ identity matrices and O is a zero matrix, such that $M_{\hat{S}}M_u = \begin{bmatrix} A & O \\ C & CA^{-1}B + D \end{bmatrix}$ is in the form of Fig. 1.

The following are the steps against UOV. We find M_u first. We let M_i be a matrix like in Fig. 2, M_u can be expressed as $M_{v+1}M_{v+2} \cdots M_{v+o}$. And the first v elements in i -th column of $M_{\hat{S}}M_i$ are all 0's. Therefore, the coefficients of x_i^2 in $P \circ (M_{\hat{S}}M_i x + V)$ are all 0's. We then get m quadratic equations in v variables.

$$M_i = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & u_{1,i} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & u_{v,i} & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Fig. 2. A Matrix Form of M_i

We thus reduce finding M_i to solving the m quadratic equations in v variables. After repeating o times, we can get M_u and find the weak key of any private key of UOV. The complexity is $o \times C_{XL}(m, v)$.

After finding M_u , we apply M_u to the public key $\hat{P} \circ \hat{S} \circ M_u = \hat{P} \circ (M_{\hat{S}}M_u x + V_{\hat{S}})$. From Definition 1, $M_{\hat{S}}M_u$ is the weak key of UOV. $(\hat{P} \circ \hat{S} \circ M_u, I)$ is another private key of UOV.

If we have a message, \mathbf{Msg} . We can get its signature, \mathbf{Sig} , by $(\hat{P} \circ \hat{S} \circ M_u, I)$. Therefore, $H(\mathbf{Msg}) = \hat{P} \circ \hat{S} \circ M_u(\mathbf{Sig})$, where H is a publicly agreed hash function. It is easy to get $M_u(\mathbf{Sig})$, which is one of the signatures by $\hat{P} \circ \hat{S}$.

Lemma 2. *If A of $M_{\hat{S}}$ is an invertible matrix, we can find a transformation, M_u , such that $M_{\hat{S}}M_u$ is the weak key of UOV.*

A has a high probability to be an invertible matrix. Even if A is not an invertible matrix, we can find a permutation matrix such that $M_{\hat{S}}M' = \begin{bmatrix} A & B \\ C & D \end{bmatrix} M' = \begin{bmatrix} A' & B' \\ C' & D' \end{bmatrix}$ and A' is an invertible matrix. $\hat{P} \circ \hat{S}$ and $\hat{P} \circ \hat{S} \circ M'$ are similar keys [7]. Therefore, we can still get one of the signatures by $\hat{P} \circ \hat{S}$.

Lemma 3. *If A of $M_{\hat{S}}$ is not an invertible matrix, we first find a permutation matrix, M' , such that $M_{\hat{S}}M'M_u$ is the weak key of UOV.*

If (P, S) and (\hat{P}, \hat{S}) are the equivalent keys of UOV [12] and (\hat{P}, \hat{S}) is the weak key of UOV, we can find a matrix such that (P, S) becomes a weak key of UOV. The time complexities depend on the MQ schemes.

3.3 Discussion of the Number of Equations and Its Security

In the original UOV scheme, $m = o$ and $n = v + o$, the attack of our scheme is not better than solving variables directly, i.e. XL algorithm. The complexities are $C_{XL}(m, n) = C_{XL}(o, o)$ and $o \times C_{XL}(m, v) = o \times C_{XL}(o, o)$ respectively.

However when the number of equations is more than the number of vinegar variables, our method is better than using XL directly. For example, the key of TRMS is a variant of UOV, although their concepts are different. The parameters in the current version of TRMS are $m = 20, n = 28, v = 19, o = 9$. The complexities are $C_{XL}(m, n) = C_{XL}(20, 28) = C_{XL}(20, 20)$ and $o \times C_{XL}(m, v) = 9 \times C_{XL}(20, 19)$ respectively. Our method is better than applying XL directly.

In multivariate signature schemes, the number of the variables is more than the number of the polynomials. Consequently $C_{XL}(m, n)$ is similar to $C_{XL}(m, m)$. When o and v are fixed, the security strength will be raised if m is larger. However, when the number of the equations is more than v , $o \times C_{XL}(m, v)$ is smaller than $C_{XL}(m, n)$. In our experiment, the best choice of m is almost v .

4 Our Attack for the Multi-layer UOV

4.1 The Multi-layer UOV

The multi-layer UOV is first described in [5]. The idea is to classify P to several sets. Each set is a independent kernel form of UOV. Let l be the number of layers, m_i is the number of equations and v_i and o_i are the numbers of vinegar

and oil variables in the i -th layer. The number of vinegar variables is the sum of the numbers of vinegar and oil variables in last layer, $v_{i+1} = v_i + o_i$ and $m_{i+1} = m_i + o_{i+1}$, for $1 \leq i < l$.

Rainbow [5] is a kind of the multi-layer UOV. Its parameters are $l = 4$, $(m_1, v_1, o_1) = (6, 6, 6)$, $(m_2, v_2, o_2) = (11, 12, 5)$, $(m_3, v_3, o_3) = (16, 17, 5)$ and $(m_4, v_4, o_4) = (27, 22, 11)$.

Since Rainbow is a variant of UOV, we can apply the attack for UOV. Its complexity is $C_{UOV}(q, v, o) = C_{UOV}(256, 22, 11) = 256^{22-11-1} \times 11^4 > 2^{93}$.

4.2 Separation of the Layers

We apply our method described in Section 3 to separate the layers. If the matrix A , defined in Section 3.2, is invertible, we first transform the private key to the weak key. We can separate the outermost layer of multi-layer UOV in complexity $o_4 \times C_{XL}(m_4, v_4) = 11 \times C_{XL}(27, 22) \approx 2^{65.5}$ 3DES operations. After transferring to the weak key, only equations in the fourth layer has the terms with fourth oil variables. Therefore, we can separate the fourth layer with others. Now, we eliminate the outermost layer in the multi-layer scheme. We can repeat to take off each layer to find the isomorphic copy of the private key or use XL to find a solution only. As our experiment, when we take off the outermost layer of the multi-layer UOV, we apply XL to get a solution.

4.3 Apply to Multi-layer UOV Schemes

We list the schemes that which private key can be viewed as a multi-layer UOV. Although their concept are not the same as UOV, their private key can be treated as a multi-layer UOV. We first use our method to take off the outermost layer UOV and then use XL to get a solution of the rest equations.

MFE [14] is a variate of TRMC, however their private key can be viewed as 2 layers UOV, $l = 2$, $(m_1, v_1, o_1) = (20, 16, 16)$ and $(m_2, v_2, o_2) = (60, 32, 16)$. The complexity to take off the outermost layer is $o_2 \times C_{XL}(m_2, v_2) = 16 \times C_{XL}(60, 32) \approx 2^{59}$ 3DES operations. And the rest is 20 equations in 16 variables.

Rainbow [5] is a standard of the multi-layer UOV. As we described in previous subsections, its has 4 layers. But we can treat it as 2 layers, $l = 2$, $(m_1, v_1, o_1) = (16, 17, 5)$ and $(m_2, v_2, o_2) = (27, 22, 11)$. After taking off the second layer, the rest is 16 equations in 22 variables. We can attack it with XL.

TRMS [13] and TRMC are based tractable rational maps. Its private key still can be viewed as 2 layer UOV, $l = 2$, $(m_1, v_1, o_1) = (11, 13, 6)$ and $(m_2, v_2, o_2) = (20, 19, 9)$. The complexity to take off the outermost layer is $o_2 \times C_{XL}(m_2, v_2) = 9 \times C_{XL}(20, 19) \approx 2^{67.6}$ 3DES operations. And the rest is 11 equations in 19 variables.

TTS [16] is built on a combination of the Oil and Vinegar and Triangular ideas. We treat its private key as 2 layer UOV, $l = 2, (m_1, v_1, o_1) = (11, 12, 7)$ and $(m_2, v_2, o_2) = (20, 19, 9)$. The complexity to take off the outermost layer is $o_2 \times C_{XL}(m_2, v_2) = 9 \times C_{XL}(20, 19) \approx 2^{67.6}$ 3DES operations. And the rest is 11 equations in 19 variables.

We analyze the security strength with Equ. (1) and (2) in Section 2.4. We are informed by Prof. Bo-Yin Yang and Jiun-Ming Chen that the parameters in Equ. (2) may be too optimal. On the contrary, the estimates of Equ. (2) only have a bit of overestimate in our experiment. Our experiment data is in Table 1. Therefore, we still use the formula in Equ. (2) to analyze the security strength.

Table 1. Estimations and Running Time of Equation 2

$C_{XL}(m, n)$	Estimate by Equ. (2)	Running time of our experiment in cycle numbers in \log_2 (in second)
$C_{XL}(10, 8)$	32.07	32.53(1.8)
$C_{XL}(11, 9)$	33.81	35.11(10.79)
$C_{XL}(12, 10)$	38.33	39.11(173.119)

When attacking current version of these schemes, first we try to get the weak key, and then use XL to find a solution to its rest equations. Here we tabulate the complexity estimate in Table 2.

Table 2. Complexity estimate of Rainbow, MFE, TRMS, TTS in \log_2

Scheme	taking off outermost layer	XL to the rest	forging a signature
Rainbow	65.5	60.5	65.5
MFE	59	44.4	59
TRMS	67.6	43.8	67.6
TTS	67.6	43.8	67.6

5 Conclusion

There are two attacks against UOV. One is extended from the attack of the "Balanced Oil and Vinegar" signature. The other is to solve the equations directly.

In this paper, we define the weak key of UOV and propose a scheme to transfer to the weak key from the public key. Our scheme is suitable for the case $m > v$. When $v \geq m > o$, applying XL directly can get better performance.

We apply our scheme to the multi-layer UOV applications. First we take off the outermost layer and then use XL to get a solution to the rest equations. We also use the complexity estimates of [15] and implement to check the parameters of these estimates. Our result shows that the current version of Rainbow, MFE, TRMS and TTS is not as secure as they are claimed to be.

References

1. Gwéno   Ars, Jean-Charles Faug  re, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita, *Comparison Between XL and Gr  bner Basis Algorithms*, Advances in Cryptology — ASIACRYPT 2004, Lecture Notes in Computer Science 3329, Springer-Verlag (2004) pp. 338-353.
2. An Braeken, Christopher Wolf and Bart Preneel, *A Study of the Security of Unbalanced Oil and Vinegar Signature Schemes*, The Cryptographers' Track at the RSA Conference 2005, Lecture Notes in Computer Science 3376, Springer-Verlag (2005) pp. 29-43, extended version: <http://eprint.iacr.org/2004/222/>.
3. Nicolas Courtois, Magnus Daum, and Patrick Felke, *On the security of HFE, HFEv and Quartz*, Public Key Cryptography — PKC 2003, Lecture Notes in Computer Science 2567, Springer-Verlag (2002) pp. 337-350.
4. Nicolas Courtois, Alexander Klimov, Jacques Patarin and Adi Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Advances in Cryptology — EUROCRYPT 2000, Lecture Notes in Computer Science 1807, Springer-Verlag (2000) pp. 392-407.
5. Jintai Ding and Dieter Schmidt, *Rainbow, a New Multivariable Polynomial Signature Scheme*, Applied Cryptography and Network Security — ACNS 2005, Lecture Notes in Computer Science 3531, Springer-Verlag (2005) pp. 164-175.
6. Michael R. Garay and David S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness*, W.H. Freeman and Company(1979), pp. 251.
7. Yuh-Hua Hu, Lih-Chung Wang, Chun-Yen Chou and Feipei Lai, *Similar Keys of Multivariate Quadratic Public Key Cryptosystems*, Cryptology and Network Security — CANS 2005, Lecture Notes in Computer Science 3810, Springer-Verlag (2005) pp. 244-257.
8. Aviad Kipnis and Adi Shamir, *Cryptanalysis of the Oil & Vinegar Signature Scheme*, Advances in Cryptology — CRYPTO'98, Lecture Notes in Computer Science 1462, Springer-Verlag (1998) pp. 257-267.
9. Aviad Kipnis, Jacques Patarin and Louis Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, Advances in Cryptology — CRYPTO'99, Lecture Notes in Computer Science 1592, Springer-Verlag (1999) pp. 206-222.
10. Jacques Patarin, *The Oil and Vinegar Algorithm for Signatures*, presented at the Dagstuhl Workshop on Cryptography, September 97.
11. Christopher Wolf, An Braeken and Bart Preneel, *Efficient Cryptanalysis of RSE(2)PKC and RSSE(2)PKC*, Security in Communication Networks, 4th International Conference, SCN 2004, Lecture Notes in Computer Science 3352, Springer-Verlag (2005) pp. 294-309, extended version: <http://eprint.iacr.org/2004/237/>.
12. Christopher Wolf and Bart Preneel, *Superfluous keys in Multivariate Quadratic asymmetric systems*, Public Key Cryptography — PKC 2005, Lecture Notes in Computer Science 3386, Springer-Verlag (2005) pp. 275-287, extended version: <http://eprint.iacr.org/2004/361/>.
13. Lih-Chung Wang, Yuh-Hua Hu, Feipei Lai, Chun-Yen Chou and Bo-Yin Yang, *Tractable Rational Map Signature*, Public Key Cryptography — PKC 2005, Lecture Notes in Computer Science 3386, Springer-Verlag (2005) pp. 244-257.
14. Lih-Chung Wang, Bo-Yin Yang, Yuh-Hua Hu and Feipei Lai, *A "Medium-Field" Multivariate Public-Key Encryption Scheme*, The Cryptographers' Track at the RSA Conference 2006, Lecture Notes in Computer Science 3860, Springer-Verlag (2006) pp. 132-149.

15. Bo-Yin Yang and Jiun-Ming Chen, *All in the XL Family: Theory and Practice*, Information Security and Cryptology — ICISC 2004, Lecture Notes in Computer Science 3506, Springer-Verlag (2004) pp. 67-86.
16. Bo-Yin Yang and Jiun-Ming Chen, *Building Secure Tame-like Multivariate Public-Key Cryptosystems The New TTS*, Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Lecture Notes in Computer Science 3574, Springer-Verlag (2005) pp. 518-531.

TRIVIUM: A Stream Cipher Construction Inspired by Block Cipher Design Principles*

Christophe De Cannière^{1,2}

¹ IAIK Krypto Group, Graz University of Technology
Inffeldgasse 16A, A-8010 Graz, Austria
`christophe.decanniere@iaik.tugraz.at`

² Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium

Abstract. In this paper, we propose a new stream cipher construction based on block cipher design principles. The main idea is to replace the building blocks used in block ciphers by equivalent stream cipher components. In order to illustrate this approach, we construct a very simple synchronous stream cipher which provides a lot of flexibility for hardware implementations, and seems to have a number of desirable cryptographic properties.

1 Introduction

In the last few years, widely used stream ciphers have started to be systematically replaced by block ciphers. An example is the A5/1 stream cipher used in the GSM standard. Its successor, A5/3, is a block cipher. A similar shift took place with wireless network standards. The security mechanism specified in the original IEEE 802.11 standard (called ‘wired equivalent privacy’ or WEP) was based on the stream cipher RC4; the newest standard, IEEE 802.11i, makes use of the block cipher AES.

The declining popularity of stream ciphers can be explained by different factors. The first is the fact that the security of block ciphers seems to be better understood. Over the last decades, cryptographers have developed a rather clear vision of what the internal structure of a secure block cipher should look like. This is much less the case for stream ciphers. Stream ciphers proposed in the past have been based on very different principles, and many of them have shown weaknesses. A second explanation is that efficiency, which has been the traditional motivation for choosing a stream cipher over a block cipher, has ceased to be a decisive factor in many applications: not only is the cost of computing power rapidly decreasing, today’s block ciphers are also significantly more efficient than their predecessors.

* The work described in this paper has been partly supported by the European Commission under contract IST-2002-507932 (ECRYPT), by the Fund for Scientific Research – Flanders (FWO), and by the Austrian Science Fund (FWF project P18138).

Still, as pointed out by the eSTREAM Stream Cipher Project, it seems that stream ciphers could continue to play an important role in those applications where high throughput remains critical and/or where resources are very restricted. This poses two challenges for the cryptographic community: first, restoring the confidence in stream ciphers, e.g., by developing simple and reliable design criteria; secondly, increasing the efficiency advantage of stream ciphers compared to block ciphers.

In this paper, we try to explore both problems. The first part of the article reviews some concepts which lie at the base of today's block ciphers (Sect. 3), and studies how these could be mapped to stream ciphers (Sects. 4–5). The design criteria derived this way are then used as a guideline to construct a simple and flexible hardware-oriented stream cipher in the second part (Sect. 6).

2 Security and Efficiency Considerations

Before devising a design strategy for a stream cipher, it is useful to first clearly specify what we expect from it. Our aim in this paper is to design hardware-oriented binary additive stream ciphers which are both efficient and secure. The following sections briefly discuss what this implies.

2.1 Security

The additive stream cipher which we intend to construct takes as input a k -bit secret key K and an n -bit IV. The cipher is then requested to generate up to 2^d bits of key stream $z_t = S_K(IV, t)$, $0 \leq t < 2^d$, and a bitwise exclusive OR of this key stream with the plaintext produces the ciphertext. The security of this additive stream cipher is determined by the extent to which it mimics a one-time pad, i.e., it should be hard for an adversary, who does not know the key, to distinguish the key stream generated by the cipher from a truly random sequence. In fact, we would like this to be as hard as we can possibly ask from a cipher with given parameters k , n , and d . This leads to a criterion called K -security [1], which can be formulated as follows:

Definition 1. *An additive stream cipher is called K -secure if any attack against this scheme would not have been significantly more difficult if the cipher had been replaced by a set of 2^k functions $S_K : \{0, 1\}^n \times \{0, \dots, 2^d - 1\} \rightarrow \{0, 1\}$, uniformly selected from the set of all possible functions.*

The definition assumes that the adversary has access to arbitrary amounts of key stream, that he knows or can choose the a priori distribution of the secret key, that he can impose relations between different secret keys, etc.

Attacks against stream ciphers can be classified into two categories, depending on what they intend to achieve:

- *Key recovery attacks*, which try to deduce information about the secret key by observing the key stream.

- *Distinguishing attacks*, the goal of which is merely to detect that the key stream bits are not completely unpredictable.

Owing to their weaker objective, distinguishing attacks are often much easier to apply, and consequently harder to protect against. Features of the key stream that can be exploited by such attacks include periodicity, dependencies between bits at different positions, non-uniformity of distributions of bits or words, etc. In this paper we will focus in particular on linear correlations, as it appeared to be the weakest aspect in a number of recent stream cipher proposals such as SOBER-*tw* [2] and SNOW 1.0 [3]. Our first design objective will be to keep the largest correlations below safe bounds. Other important properties, such as a sufficiently long period, are only considered afterwards. Note that this approach differs from the way LFSR or T-function based schemes are constructed. The latter are typically designed by maximizing the period first, and only then imposing additional requirements.

2.2 Efficiency

In order for a stream cipher to be an attractive alternative to block ciphers, it must be efficient. In this paper, we will be targeting hardware applications, and a good measure for the efficiency of a stream cipher in this environment is the number of key stream bits generated per cycle per gate.

There are two ways to obtain an efficient scheme according to this measure. The first approach is illustrated by A5/1, and consists in minimizing the number of gates. A5/1 is extremely compact in hardware, but it cannot generate more than one bit per cycle. The other approach, which was chosen by the designers of PANAMA [4], is to dramatically increase the number of bits per cycle. This allows to reduce the clock frequency (and potentially also the power consumption) at the cost of an increased gate count. As a result, PANAMA is not suited for environments with very tight area constraints. Similarly, designs such as A5/1 will not perform very well in systems which require fast encryption at a low clock frequency. One of the objectives of this paper is to design a flexible scheme which performs reasonably well in both situations.

3 How Block Ciphers Are Designed

As explained above, the first requirement we impose on the construction is that it generates key streams without exploitable linear correlations. This problem is very similar to the one faced by block cipher designers. Hence, it is natural to attempt to borrow some of the techniques used in the block cipher world. The ideas relevant to stream ciphers are briefly reviewed in the following sections.

3.1 Block Ciphers and Linear Characteristics

An important problem in the case of block ciphers is that of restricting linear correlations between input and output bits in order to thwart linear cryptanalysis [5]. More precisely, let P be any plaintext block and C the corresponding ciphertext under a fixed secret key, then any linear combination of bits

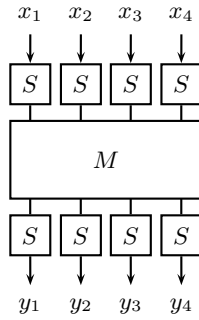


Fig. 1. Three layers of a block cipher

$$\Gamma_P^T \cdot P + \Gamma_C^T \cdot C,$$

where the column vectors Γ_P and Γ_C are called *linear masks*, should be as balanced as possible. That is, the correlation

$$c = 2 \cdot \frac{|\{P \mid \Gamma_P^T \cdot P = \Gamma_C^T \cdot C\}|}{|\{P\}|} - 1$$

has to be close to 0 for any Γ_P and Γ_C . The well-established way to achieve this consists in alternating two operations. The first splits blocks into smaller words which are independently fed into nonlinear substitution boxes (S-boxes); the second step recombines the outputs of the S-boxes in a linear way in order to ‘diffuse’ the nonlinearity. The result, called a substitution-permutation network, is depicted in Fig. 1.

In order to estimate the strength of a block cipher against linear cryptanalysis, one will typically compute bounds on the correlation of *linear characteristics*. A linear characteristic describes a possible path over which a correlation might propagate through the block cipher. It is a chain of linear masks, starting with a plaintext mask and ending with a ciphertext mask, such that every two successive masks correspond to a nonzero correlation between consecutive intermediate values in the cipher. The total correlation of the characteristic is then estimated by multiplying the correlations of all separate steps (as dictated by the so-called Piling-up Lemma).

3.2 Branch Number

Linear diffusion layers, which can be represented by a matrix multiplication $Y = M \cdot X$, do not by themselves contribute in reducing the correlation of a characteristic. Clearly, it suffices to choose $\Gamma_X = M^T \cdot \Gamma_Y$, where M^T denotes the transpose of M , in order to obtain perfectly correlating linear combinations of X and Y :

$$\Gamma_Y^T \cdot Y = \Gamma_Y^T \cdot MX = (M^T \Gamma_Y)^T \cdot X = \Gamma_X^T \cdot X.$$

However, diffusion layers play an important indirect role by forcing characteristics to take into account a large number of nonlinear S-boxes in the neighboring

layers (called *active* S-boxes). A useful metric in this context is the *branch number* of M .

Definition 2. *The branch number of a linear transformation M is defined as*

$$\mathcal{B} = \min_{\Gamma_Y \neq 0} [w_h(\Gamma_Y) + w_h(M^T \Gamma_Y)],$$

where $w_h(\Gamma)$ represents the number of nonzero words in the linear mask Γ .

The definition above implies that any linear characteristic traversing the structure shown in Fig. 1 activates at least \mathcal{B} S-boxes. The total number of active S-boxes throughout the cipher multiplied by the maximal correlation over a single S-box gives an upper bound for the correlation of the characteristic.

The straightforward way to minimize this upper bound is to maximize the branch number \mathcal{B} . It is easy to see that \mathcal{B} cannot exceed $m + 1$, with m the number of words per block. Matrices M that satisfy this bound (known as the Singleton bound) can be derived from the generator matrices of maximum distance separable (MDS) block codes.

Large MDS matrices are expensive to implement, though. Therefore, it is often more efficient to use smaller matrices, with a relatively low branch number, and to connect them in such a way that linear patterns with a small number of active S-boxes cannot be chained together to cover the complete cipher. This was the approach taken by the designers of RIJNDAEL [6].

4 From Blocks to Streams

In this section, we try to adapt the concepts described above to a system where the data is not processed in blocks, but rather as a stream.

Since data enters the system one word at a time, each layer of S-boxes in Fig. 1 can be replaced by a single S-box which substitutes individual words as they arrive. A general m th-order linear filter can take over the task of the diffusion matrix. The new system is represented in Fig. 2, where D denotes the delay operator (usually written as z^{-1} in signal processing literature), and f and g are linear functions.

4.1 Polynomial Notation

Before analyzing the properties of this construction, we introduce some notations. First, we adopt the common convention to represent streams of words x_0, x_1, x_2, \dots as polynomials with coefficients in the finite field:

$$x(D) = x_0 + x_1 D + x_2 D^2 + \dots$$

The rationale for this representation is that it simplifies the expression for the input/output relation of the linear filter, as shown in the following equation:

$$y(D) = \frac{f(D)}{g(D)} \cdot [x(D) + x^0(D)] + y^0(D). \tag{1}$$

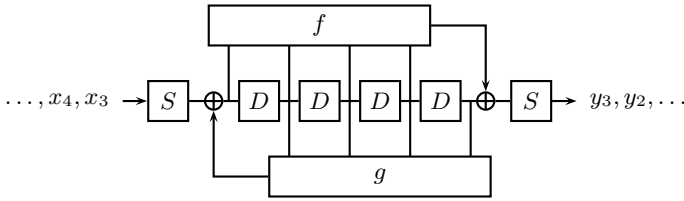


Fig. 2. Stream equivalent of Fig. 1

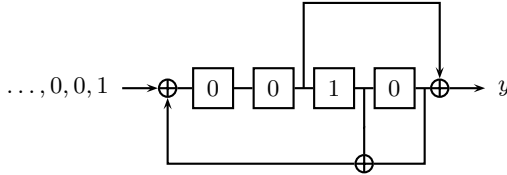


Fig. 3. A 4th-order linear filter

The polynomials f and g describe the feedforward and feedback connections of the filter. They can be written as

$$f(D) = D^m \cdot (f_m D^{-m} + \dots + f_1 D^{-1} + 1) ,$$

$$g(D) = 1 + g_1 D + g_2 D^2 + \dots + g_m D^m .$$

The Laurent polynomials x^0 and y^0 represent the influence of the initial state s^0 , and are given by $x^0 = D^{-m} \cdot (s^0 \cdot g \bmod D^m)$ and $y^0 = D^{-m} \cdot (s^0 \cdot f \bmod D^m)$.

Example 1. The 4th-order linear filter depicted in Fig. 3 is specified by the polynomials $f(D) = D^4 \cdot (D^{-2} + 1)$ and $g(D) = 1 + D^3 + D^4$. Suppose that the delay elements are initialized as shown in the figure, i.e., $s^0(D) = D$. Knowing s^0 , we can compute $x^0(D) = D^{-3}$ and $y^0(D) = D^{-1}$. Finally, using (1), we find the output stream corresponding to an input consisting, for example, of a single 1 followed by 0's (i.e., $x(D) = 1$):

$$y(D) = \frac{D^{-1} + D + D^2 + D^4}{1 + D^3 + D^4} + D^{-1}$$

$$= D + D^3 + D^5 + D^6 + D^7 + D^8 + D^{12} + D^{15} + D^{16} + D^{18} + \dots$$

4.2 Linear Correlations

In order to study correlations in a stream-oriented system we need a suitable way to manipulate linear combinations of bits in a stream. It will prove convenient to represent them as follows:

$$\text{Tr} \left[[\gamma_x(D^{-1}) \cdot x(D)]_0 \right] .$$

The operator $[\cdot]_0$ returns the constant term of a polynomial, and $\text{Tr}(\cdot)$ denotes the trace to $\text{GF}(2)$.¹ The coefficients of γ_x , called *selection polynomial*, specify which words of x are involved in the linear combination. In order to simplify expressions later on we also introduce the notation $\gamma^*(D) = \gamma(D^{-1})$. The polynomial γ^* is called the *reciprocal* polynomial of γ .

As before, the correlation between x and y for a given pair of selection polynomials is defined as

$$c = 2 \cdot \frac{|\{(x, s^0) \mid \text{Tr}[[\gamma_x^* \cdot x]_0] = \text{Tr}[[\gamma_y^* \cdot y]_0]\}|}{|\{(x, s^0)\}|} - 1,$$

where $\deg x \leq \max(\deg \gamma_x, \deg \gamma_y)$.

4.3 Propagation of Selection Polynomials

Let us now analyze how correlations propagate through the linear filter. For each selection polynomial γ_x at the input, we would like to determine a polynomial γ_y at the output (if it exists) such that the corresponding linear combinations are perfectly correlated, i.e.,

$$\text{Tr}[[\gamma_x^* \cdot x]_0] = \text{Tr}[[\gamma_y^* \cdot y]_0], \quad \forall x, s^0.$$

If this equation is satisfied, then this will still be the case after replacing x by $x' = x + x^0$ and y by $y' = y + y^0$, since x^0 and y^0 only consist of negative powers, none of which can be selected by γ_x or γ_y . Substituting (1), we find

$$\text{Tr}[[\gamma_x^* \cdot x']_0] = \text{Tr}[[\gamma_y^* \cdot f/g \cdot x']_0], \quad \forall x, s^0,$$

which implies that $\gamma_x^* = \gamma_y^* \cdot f/g$. In order to get rid of negative powers, we define $f^* = D^m \cdot f$ and $g^* = D^m \cdot g$ (note the subtle difference between both stars), and obtain the equivalent relation

$$\gamma_y = g^* / f^* \cdot \gamma_x. \quad (2)$$

Note that neither of the selection polynomials γ_x and γ_y can have an infinite number of nonzero coefficients (if it were the case, the linear combinations would be undefined). Hence, they have to be of the form

$$\gamma_x = q \cdot f^* / \text{gcd}(f^*, g^*) \quad \text{and} \quad \gamma_y = q \cdot g^* / \text{gcd}(f^*, g^*), \quad (3)$$

with $q(D)$ an arbitrary polynomial.

Example 2. For the linear filter in Fig. 3, we have that $f^*(D) = 1 + D^2$ and $g^*(D) = D^4 \cdot (D^{-4} + D^{-3} + 1)$. In this case, f^* and g^* are coprime, i.e., $\text{gcd}(f^*, g^*) = 1$. If we arbitrarily choose $q(D) = 1 + D$, we obtain a pair of selection polynomials

$$\gamma_x(D) = 1 + D + D^2 + D^3 \quad \text{and} \quad \gamma_y(D) = 1 + D^2 + D^4 + D^5.$$

By construction, the corresponding linear combinations of input and output bits satisfy the relation

$$\text{Tr}(x_0 + x_1 + x_2 + x_3) = \text{Tr}(y_0 + y_2 + y_4 + y_5), \quad \forall x, s^0.$$

¹ The trace from $\text{GF}(2^n)$ to $\text{GF}(2)$ is defined as $\text{Tr}(a) = a + a^2 + a^4 + \dots + a^{2^{n-1}}$.

4.4 Branch Number

The purpose of the linear filter, just as the diffusion layer of a block cipher, will be to force linear characteristics to pass through as many active S-boxes as possible. Hence, it makes sense to define a branch number here as well.

Definition 3. *The branch number of a linear filter specified by the polynomials f and g is defined as*

$$\begin{aligned} \mathcal{B} &= \min_{\gamma_x \neq 0} [\text{w}_h(\gamma_x) + \text{w}_h(g^*/f^* \cdot \gamma_x)] \\ &= \min_{q \neq 0} [\text{w}_h(q \cdot f^*/\text{gcd}(f^*, g^*)) + \text{w}_h(q \cdot g^*/\text{gcd}(f^*, g^*))], \end{aligned}$$

where $\text{w}_h(\gamma)$ represents the number of nonzero coefficients in the selection polynomial γ .

From this definition we immediately obtain the following upper bound on the branch number

$$\mathcal{B} \leq \text{w}_h(f^*) + \text{w}_h(g^*) \leq 2 \cdot (m + 1). \quad (4)$$

Filters for which this bound is attained can be derived from MDS convolutional $(2, 1, m)$ -codes [7]. For example, one can verify that the 4th-order linear filter over $\text{GF}(2^8)$ with

$$\begin{aligned} f(D) &= D^4 \cdot (02_x D^{-4} + D^{-3} + D^{-2} + 02_x D^{-1} + 1), \\ g(D) &= 1 + 03_x D + 03_x D^2 + D^3 + D^4, \end{aligned}$$

has a branch number of 10. Note that this example uses the same field polynomial as RIJNDAEL, i.e., $x^8 + x^4 + x^3 + x + 1$.

5 Constructing a Key Stream Generator

In the previous section, we introduced S-boxes and linear filters as building blocks, and presented some tools to analyze how they interact. Our next task is to determine how these components can be combined into a key stream generator. Again, block ciphers will serve as a source of inspiration.

5.1 Basic Construction

A well-known way to construct a key stream generator from a block cipher is to use the cipher in output feedback (OFB) mode. This mode of operation takes as input an initial data block (called initial value or IV), passes it through the block cipher, and feeds the result back to the input. This process is iterated and the consecutive values of the data block are used as key stream. We recall that the block cipher itself typically consists of a sequence of rounds, each comprising a layer of S-boxes and a linear diffusion transformation.

By taking the very same approach, but this time using the stream cipher components presented in Sect. 4, we obtain a construction which, in its simplest

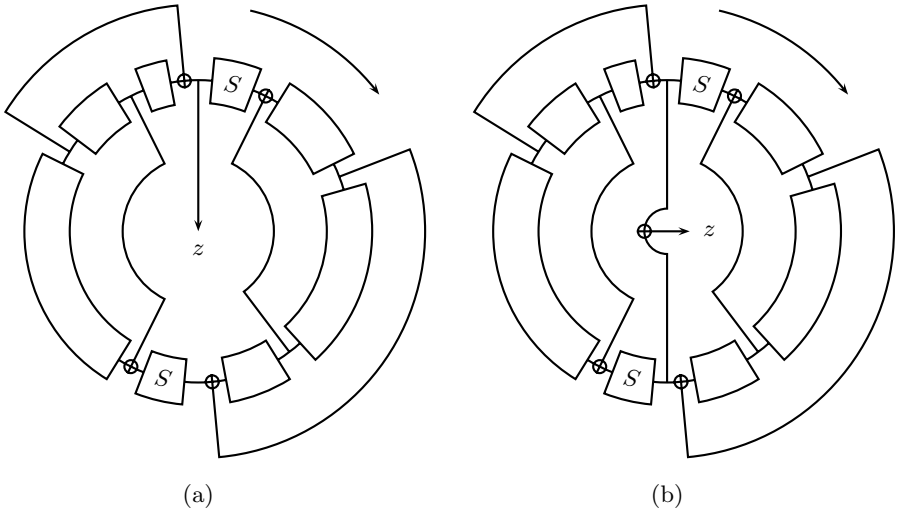


Fig. 4. Two-round key stream generators

form, might look like Fig. 4(a). The figure represents a key stream generator consisting of two ‘rounds’, where each round consists of an S-box followed by a very simple linear filter. Data words traverse the structure in clockwise direction, and the output of the second round, which also serves as key stream, is fed back to the input of the first round.

While the scheme proposed above has some interesting structural similarities with a block cipher in OFB mode, there are important differences as well. The most fundamental difference comes from the fact that linear filters, as opposed to diffusion matrices, have an internal state. Hence if the algorithm manages to keep this state (or at least parts of it) secret, then this eliminates the need for a separate key addition layer (another important block cipher component, which we have tacitly ignored so far).

5.2 Analysis of Linear Characteristics

As stated before, the primary goal in this paper is to construct a scheme which generates a stream of seemingly uncorrelated bits. More specifically, we would like the adversary to be unable to detect any correlation between linear combinations of bits at different positions in the key stream. In the following sections, we will see that the study of linear characteristics provides some guidance on how to design the components of our scheme in order to reduce the magnitude of these correlations.

Applying the tools from Sect. 4 to the construction in Fig. 4(a), we can easily derive some results on the existence of low-weight linear characteristics. The term ‘low-weight’ in this context refers to a small number of active S-boxes. Since we are interested in correlations which can be detected by an adversary, we need both ends of the characteristic to be accessible from the key stream. In order to construct such characteristics, we start with a selection polynomial γ_u at the input of the first round, and analyze how it might propagate through the cipher.

First, the characteristic needs to cross an S-box. The S-box preserves the positions of the non-zero coefficients of γ_u , but might modify their values. For now, however, let us only consider characteristics for which the values are preserved as well. Under this assumption and using (2), we can compute the selection polynomials γ_v and γ_w at the input and the output of the second round:

$$\gamma_v = g_1^*/f_1^* \cdot \gamma_u \quad \text{and} \quad \gamma_w = g_2^*/f_2^* \cdot \gamma_v.$$

Since all three polynomials γ_u , γ_v , and γ_w need to be finite, we have that

$$\gamma_u = q \cdot f_1^* f_2^*/d, \quad \gamma_v = q \cdot g_1^* f_2^*/d, \quad \text{and} \quad \gamma_w = q \cdot g_1^* g_2^*/d,$$

with $d = \gcd(f_1^* f_2^*, g_1^* f_2^*, g_1^* g_2^*)$ and q an arbitrary polynomial. Note that since both γ_u and γ_w select bits from the key stream z , they can be combined into a single polynomial $\gamma_z = \gamma_u + \gamma_w$.

The number of S-boxes activated by a characteristic of this form is given by $\mathcal{W} = w_h(\gamma_u) + w_h(\gamma_v)$. The minimum number of active S-boxes over this set of characteristics can be computed with the formula

$$\mathcal{W}_{\min} = \min_{q \neq 0} [w_h(q \cdot f_1^* f_2^*/d) + w_h(q \cdot g_1^* f_2^*/d)],$$

from which we derive that

$$\mathcal{W}_{\min} \leq w_h(f_1^* f_2^*) + w_h(g_1^* f_2^*) \leq w_h(f_1^*) \cdot w_h(f_2^*) + w_h(g_1^*) \cdot w_h(f_2^*).$$

Applying this bound to the specific example of Fig. 4(a), where $w_h(f_i^*) = w_h(g_i^*) = 2$, we conclude that there will always exist characteristics with at most 8 active S-boxes, no matter where the taps of the linear filters are positioned.

5.3 An Improvement

We will now show that this bound can potentially be doubled by making the small modification shown in Fig. 4(b). This time, each non-zero coefficient in the selection polynomial at the output of the key stream generator needs to propagate to both the upper and the lower part of the scheme. By constructing linear characteristics in the same way as before, we obtain the following selection polynomials:

$$\gamma_u = q \cdot \frac{f_1^* f_2^* + f_1^* g_2^*}{d}, \quad \gamma_v = q \cdot \frac{f_1^* f_2^* + g_1^* f_2^*}{d}, \quad \text{and} \quad \gamma_z = q \cdot \frac{f_1^* f_2^* + g_1^* g_2^*}{d},$$

with $d = \gcd(f_1^* f_2^* + f_1^* g_2^*, f_1^* f_2^* + g_1^* f_2^*, f_1^* f_2^* + g_1^* g_2^*)$. The new upper bounds on the minimum number of active S-boxes are given by

$$\begin{aligned} \mathcal{W}_{\min} &\leq w_h(f_1^* f_2^* + f_1^* g_2^*) + w_h(f_1^* f_2^* + g_1^* f_2^*) \\ &\leq 2 \cdot w_h(f_1^*) \cdot w_h(f_2^*) + w_h(f_1^*) \cdot w_h(g_2^*) + w_h(g_1^*) \cdot w_h(f_2^*), \end{aligned}$$

or, in the case of Fig. 4(b), $\mathcal{W}_{\min} \leq 16$. In general, if we consider extensions of this scheme with r rounds and $w_h(f_i^*) = w_h(g_i^*) = w$, then the bound takes the form:

$$\mathcal{W}_{\min} \leq r^2 \cdot w^r. \tag{5}$$

This result suggests that it might not be necessary to use a large number of rounds, or complicated linear filters, to ensure that the number of active S-boxes in all characteristics is sufficiently large. For example, if we take $w = 2$ as before, but add one more round, the bound jumps to 72.

Of course, since the bound we just derived is an upper bound, the minimal number of active S-boxes might as well be much smaller. First, some of the product terms in $f_1^* f_2^* + f_1^* g_2^*$ or $f_1^* f_2^* + g_1^* f_2^*$ might cancel out, or there might exist a $q \neq d$ for which $w_h(\gamma_u) + w_h(\gamma_v)$ suddenly drops. These cases are rather easy to detect, though, and can be avoided during the design. A more important problem is that we have limited ourselves to a special set of characteristics, which might not necessarily include the one with the minimal number of active S-boxes. However, if the feedback and feedforward functions are sparse, and the linear filters sufficiently large, then the bound is increasingly likely to be tight. On the other hand, if the state of the generator is sufficiently small, then we can perform an efficient search for the lowest-weight characteristic without making any additional assumption.

This last approach allows to show, for example, that the smallest instance of the scheme in Fig. 4(b) for which the bound of 16 is actually attained, consists of two 11th-order linear filters with

$$\begin{aligned} f_1^*(D) &= 1 + D^{10}, & g_1^*(D) &= D^{11} \cdot (D^{-3} + 1), \\ f_2^*(D) &= 1 + D^9, & g_2^*(D) &= D^{11} \cdot (D^{-8} + 1). \end{aligned}$$

5.4 Linear Characteristics and Correlations

In the sections above, we have tried to increase the number of active S-boxes of linear characteristics. We now briefly discuss how this number affects the correlation of key stream bits. This problem is treated in several papers in the context of block ciphers (see, e.g., [6]).

We start with the observation that the minimum number of active S-boxes \mathcal{W}_{\min} imposes a bound on the correlation c_c of a linear characteristic:

$$c_c^2 \leq (c_s^2)^{\mathcal{W}_{\min}},$$

where c_s is the largest correlation (in absolute value) between the input and the output values of the S-box. The squares c_c^2 and c_s^2 are often referred to as *linear probability*, or also *correlation potential*. The inverse of this quantity is a good measure for the amount of data that the attacker needs to observe in order to detect a correlation.

What makes the analysis more complicated, however, is that many linear characteristics can contribute to the correlation of the same combination of key stream bits. This occurs in particular when the scheme operates on words, in which case there are typically many possible choices for the coefficients of the intermediate selection polynomials describing the characteristic (this effect is called *clustering*). The different contributions add up or cancel out, depending on the signs of c_c . If we now assume that these signs are randomly distributed,

then we can use the approach of [6, Appendix B] to derive a bound on the expected correlation potential of the key stream bits:

$$E(c^2) \leq (c_s^2)^{W_{\min}-n}. \quad (6)$$

The parameter n in this inequality represents the number of degrees of freedom in the choice for the coefficients of the intermediate selection polynomials.

For the characteristics propagating through the construction presented in Sect. 5.3, one will find, in non-degenerate cases, that the values of $n = r \cdot (r - 1) \cdot w^{r-1}$ non-zero coefficients can be chosen independently. Hence, for example, if we construct a scheme with $w = 2$ and $r = 3$, and if we assume that it attains the bound given in (5), then we expect the largest correlation potential to be at most $c_s^{2 \cdot 48}$. Note that this bound is orders of magnitude higher than the contribution of a single characteristic, which has a correlation potential of at most $c_s^{2 \cdot 72}$.

Remark 1. In order to derive (6), we replaced the signs of the contributing linear characteristics by random variables. This is a natural approach in the case of block ciphers, where the signs depend on the value of the secret key. In our case, however, the signs are fixed for a particular scheme, and hence they might, for some special designs, take on very peculiar values. This happens for example when $r = 2$, w is even, and all non-zero coefficients of f_i and g_i equal 1 (as in the example at the end of the previous section). In this case, all signs will be positive, and we obtain a significantly worse bound:

$$c^2 \leq (c_s^2)^{W_{\min}-2 \cdot n}.$$

6 Trivium

In this final section, we present an experimental 80-bit key stream cipher based on the approach outlined above. Because of space restrictions, we limit ourselves to a very rough sketch of some basic design ideas behind the scheme. The complete specifications of the cipher, which was submitted to the eSTREAM Stream Cipher Project under the name TRIVIUM, can be found at <http://www.ecrypt.eu.org/stream/> [8].

6.1 A Bit-Oriented Design

The main idea of TRIVIUM's design is to turn the general scheme of Sect. 5.3 into a bit-oriented stream cipher. The first motivation is that bit-oriented schemes are typically more compact in hardware. A second reason is that, by reducing the word-size to a single bit, we may hope to get rid of the clustering phenomenon which, as seen in the previous section, has a significant effect on the correlation.

Of course, if we simply apply the previous scheme to bits instead of words, we run into the problem that the only two existing 1×1 -bit S-boxes are both linear. In order to solve this problem, we replace the S-boxes by a component

which, from the point of view of our correlation analysis, behaves in the same way: an exclusive OR with an external stream of unrelated but biased random bits. Assuming that these random bits equal 0 with probability $(1 + c_s)/2$, we will find as before that the output of this component correlates with the input with correlation coefficient c_s .

The introduction of this artificial 1×1 -bit S-box greatly simplifies the correlation analysis, mainly because of the fact that the selection polynomial at the output of an S-box is now uniquely determined by the input. As a consequence, we neither need to make special assumptions about the values of the non-zero coefficients, nor to consider the effect of clustering: the maximum correlation in the key stream is simply given by the relation

$$c_{\max} = c_s^{\mathcal{W}_{\min}}. \quad (7)$$

The obvious drawback, however, is that the construction now relies on external streams of random bits, which have to be generated somehow. TRIVIUM attempts to solve this problem by interleaving three identical key stream generators, where each generator obtains streams of biased bits (with $c_s = 1/2$) by ANDing together state bits of the two other generators.

6.2 Specifying the Parameters

Let us now specify suitable parameters for each of those three identical ‘sub-generators’. Our goal is to keep all parameters as small and simple as possible, given a number of requirements.

The first requirement we impose is that the correlations in the key stream do not exceed 2^{-40} . Since each sub-generator will be fed with streams of bits having correlation coefficient $c_s = 1/2$, we can derive from (7) that a minimum weight \mathcal{W}_{\min} of at least 40 is needed. The smallest values of w and r for which this requirement could be satisfied (with a fairly large margin, in fact) are $w = 2$ and $r = 3$. As an additional requirement, we would like the minimum weight to reach the upper bound of (5) for the chosen values of w and r . In this case, this translates to the condition $\mathcal{W}_{\min} = 72$, which is fulfilled if $w_h(\gamma_u) + w_h(\gamma_v) + w_h(\gamma_w) \geq 72$ for all $q \neq 0$, where

$$\gamma_u = q \cdot \frac{f_1^* f_2^* f_3^* + f_1^* f_2^* g_3^* + f_1^* g_2^* g_3^*}{d}, \quad \gamma_v = \dots, \quad \text{etc.}$$

Although the preceding sections have almost exclusively focused on linear correlations, other security properties such as periodicity remain important. Controlling the period of the scheme is difficult because of the non-linear interaction between the sub-generators, but we can try to decrease the probability of short cycles by maximizing the periods of the individual sub-generators after turning off the streams feeding their 1×1 -bit S-boxes. The connection polynomial of these (completely linear) generators is given by $f_1^* f_2^* f_3^* + g_1^* g_2^* g_3^*$, and ideally, we would like this polynomial to be primitive. Our choice of w prevents this, though: for $w = 2$, the polynomial above is always divisible by $(D + 1)^3$. Therefore, we

just require that the remaining factor is primitive, and rely on the initialization of the state bits to avoid the few short cycles corresponding to the factor $(D+1)^3$ (see [8]).

Finally, we also impose some efficiency requirements. The first is that state bits of the sub-generators should not be used for at least 64/3 iterations, once they have been modified. This will provide the final scheme with the flexibility to generate up to 64 bits in parallel. Secondly, the length of the sub-generators should be as short as possible and a multiple of 32.

We can now exhaustively run over all possible polynomials f_1^*, \dots, g_3^* in order to find combinations for which all previous requirements are fulfilled simultaneously. Surprisingly enough, it turns out that the solution is unique:

$$\begin{aligned}
 f_1^*(D) &= 1 + D^9, & g_1^*(D) &= D^{31} \cdot (D^{-23} + 1), \\
 f_2^*(D) &= 1 + D^5, & g_2^*(D) &= D^{28} \cdot (D^{-26} + 1), \\
 f_3^*(D) &= 1 + D^{15}, & g_3^*(D) &= D^{37} \cdot (D^{-29} + 1).
 \end{aligned}$$

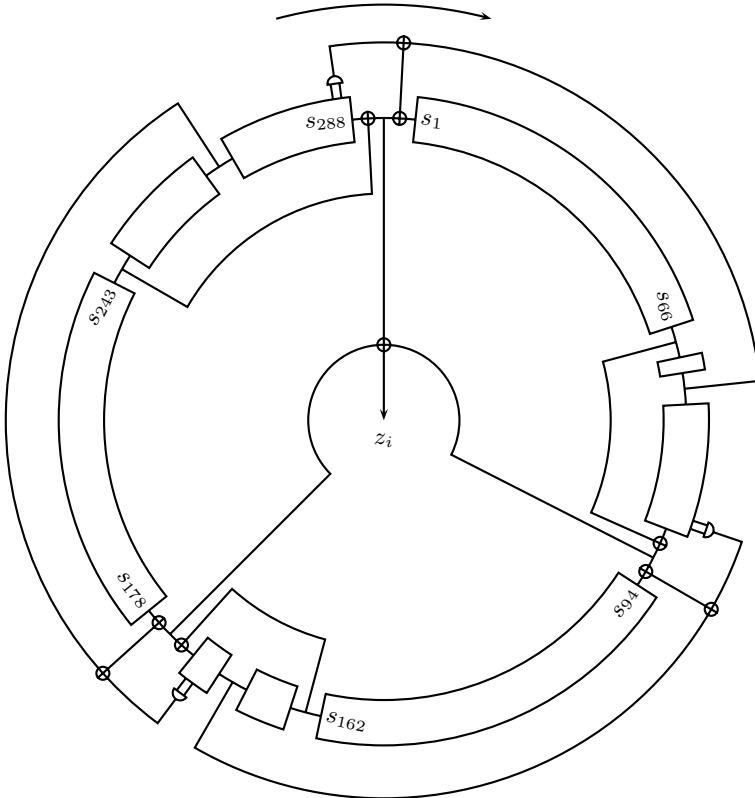


Fig. 5. TRIVIUM

In order to construct the final cipher, we interleave three of these sub-generators and interconnect them through AND-gates. Since the reasoning above does not suggest which state bits to use as inputs of the AND-gates, we simply choose to minimize the length of the wires. The resulting scheme is shown in Fig. 5. The 96 state bits $s_1, s_4, s_7, \dots, s_{286}$ belong to the first sub-generator, $s_2, s_5, s_8, \dots, s_{287}$ to the second one, etc.

6.3 Security and Efficiency Evaluation

The scheme presented above is currently being evaluated in the framework of the eSTREAM Stream Cipher Project, and to conclude this paper we briefly summarize the current status of the evaluation.

The complexities of the different attacks discovered so far are listed Table 1. The most efficient dedicated attack is a guess-and-determine attack presented by S. Khazaei [9]. However, with a time complexity of 2^{135} , it is still considerably less efficient than a generic exhaustive key search.

The hardware efficiency of TRIVIUM was independently evaluated by Gürkaynak et al. [11] and by Good et al. [12]. The first paper reports a 64-bit implementation in $0.25 \mu\text{m}$ 5-metal CMOS technology with a throughput per area ratio of $129 \text{ Gbit/s} \cdot \text{mm}^2$, which is three times higher than for any other eSTREAM candidate. The second paper presents a compact FPGA implementation with an estimated equivalent number of gates of 2682, making TRIVIUM the second most compact candidate after Grain.

Table 1. Cryptanalytical results

Attack	Time	Data	Reference
Linear distinguisher	2^{144}	2^{144}	[8]
Guess-and-determine attack	2^{195}	288	[8]
Guess-and-determine attack	2^{135}	288	[9]
Solving system of equations	2^{164}	288	[10]
Exhaustive key search	2^{80}	80	

References

1. Daemen, J.: Cipher and hash function design. Strategies based on linear and differential cryptanalysis. PhD thesis, Katholieke Universiteit Leuven (1995)
2. Hawkes, P., Rose, G.G.: Primitive specification and supporting documentation for SOBER-*tw* submission to NESSIE. In: Proceedings of the First NESSIE Workshop, NESSIE (2000)
3. Ekdahl, P., Johansson, T.: SNOW – A new stream cipher. In: Proceedings of the First NESSIE Workshop, NESSIE (2000)
4. Daemen, J., Clapp, C.S.K.: Fast hashing and stream encryption with PANAMA. In Vaudenay, S., ed.: Fast Software Encryption, FSE'98. Volume 1372 of Lecture Notes in Computer Science., Springer-Verlag (1998) 60–74

5. Matsui, M.: Linear cryptanalysis method for DES cipher. In Helleseth, T., ed.: *Advances in Cryptology – EUROCRYPT'93*. Volume 765 of *Lecture Notes in Computer Science.*, Springer-Verlag (1993) 386–397
6. Daemen, J., Rijmen, V.: *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer-Verlag (2002)
7. Rosenthal, J., Smarandache, R.: Maximum distance separable convolutional codes. *Applicable Algebra in Engineering, Communication and Computing* **10** (1999) 15–32
8. De Cannière, C., Preneel, B.: TRIVIUM — Specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030 (2005) <http://www.ecrypt.eu.org/stream>.
9. Khazaei, S.: Re: A reformulation of TRIVIUM. Posted on the eSTREAM Forum (2006) <http://www.ecrypt.eu.org/stream/phorum/read.php?1,448>.
10. Raddum, H.: Cryptanalytic results on TRIVIUM. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039 (2006) <http://www.ecrypt.eu.org/stream>.
11. Gürkaynak, F.K., Luethi, P., Bernold, N., Blattmann, R., Goode, V., Marghitola, M., Kaeslin, H., Felber, N., Fichtner, W.: Hardware evaluation of eSTREAM candidates: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, TRIVIUM, VEST, ZK-Crypt. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/015 (2006) <http://www.ecrypt.eu.org/stream>.
12. Good, T., Chelton, W., Benaissa, M.: Review of stream cipher candidates from a low resource hardware perspective. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/016 (2006) <http://www.ecrypt.eu.org/stream>.

Cryptanalysis of the Bluetooth E_0 Cipher Using OBDD's

Yaniv Shaked and Avishai Wool

School of Electrical Engineering Systems,
Tel Aviv University, Ramat Aviv 69978, Israel
shakedy@eng.tau.ac.il, yash@acm.org

Abstract. In this paper we analyze the E_0 cipher, which is the cipher used in the Bluetooth specifications. We adapted and optimized the Binary Decision Diagram attack of Krause, for the specific details of E_0 . Our method requires 128 known bits of the keystream in order to recover the initial value of the four LFSR's in the E_0 system. We describe several variants which we built to lower the complexity of the attack. We evaluated our attack against the real (non-reduced) E_0 cipher. Our best attack can recover the initial value of the four LFSR's, for the first time, with a realistic space complexity of 2^{23} (84MB RAM), and with a time complexity of 2^{87} . This attack can be massively parallelized to lower the overall time complexity. Beyond the specifics of E_0 , our work describes practical experience with BDD-based cryptanalysis, which so far has mostly been a theoretical concept.

Keywords: Stream cipher, Cryptanalysis, Bluetooth, BDD.

1 Introduction

1.1 Background

Bluetooth, a technology used for short range fast communications, has quickly spread worldwide. Bluetooth technology is used in a large set of wired and wireless devices: mobile phones, PDA's, desktop and mobile PC's, printers, digital cameras, and dozens of other devices.

Bluetooth employs a stream cipher as the data encryption mechanism. This stream cipher, E_0 , is based on 4 LFSR's (Linear Feedback Shift Registers) of different lengths, along with a non-linear blender logic (finite state machine). The keystream is xor-ed with the plaintext, to create the ciphertext, and decryption is performed in exactly the same way using the same stream used for encryption.

1.2 Related Work

A number of crypt-analytical results regarding E_0 ([JW01], [FL01], [LW05], [Kra02], [Saa00], [HN99], [EJ00], [GBM02], [LV04], [LMV05], [KS06]) have appeared over the last five years. These attacks can be organized into two classes: *Short Keystream attacks* - attacks that need at most 3,100 known keystream

bits; and *Long Keystream attacks* - attacks that require more (usually much more) known keystream. Long keystream attacks are generally not applicable within the Bluetooth settings since a maximal Bluetooth continuous frame is shorter than 3,100 bits (5 slots, $625\mu\text{sec}$ each, 1Mbit burst rate; see [Blu03], Vol 2, part B, page 59) after which Bluetooth rekeys the E_0 registers. Therefore, all long keystream attacks, except for the attack suggested in [LMV05], are applicable only if E_0 is used outside the Bluetooth system.

Short Keystream Attacks

1. D. Bleichenbacher has shown in [JW01] that an attacker can guess the initial state of the three smaller LFSR's and the non-linear blender; Then the attacker can compute the contents of the longest LFSR, (whose length is 39 bits) by "reverse engineering" it from the outputs of the other LFSR's and the blender state. This attack requires approximately 132 bits of known keystream with a computational complexity of $O(2^{100})$.
2. S. Fluhrer and S. Lucks have shown in [FL01] an optimized backtracking method of recovering the secret key with a computational complexity of $O(2^{84})$ if a 132 bits are available.
3. O. Levy and A. Wool have shown in [LW05] a uniform framework for cryptanalysis, whose best setting can recover the initial state of the LFSR's after solving $O(2^{86})$ systems of boolean linear equations.
4. The best reported short keystream attack against E_0 was suggested by Krause [Kra02] as part of a general framework. The general attack framework uses Free Binary Decision Diagram (FBDD's), a data structure that is used to represent a boolean function, for attacking LFSR-based key stream generators in general, and E_0 in particular. In his paper, Krause claims that for E_0 his attack requires $O(2^{77})$ space, and a time complexity of $O(2^{81})$, based on some quick estimations. Krause's attack is the starting point of this paper: we adapted and optimized his attack for the specifics of E_0 , and evaluated the attack's viability.

The work closest to ours was very recent recently suggested, independently, by Krause and Stegemann [KS06]. They too attempt to make BDD-based cryptanalysis practical, via a divide-and-conquer strategy. They evaluated their attacks against reduced versions of E_0 , with random feedback polynomials, and extrapolated a space complexity of $O(2^{42})$ against the real E_0 , with roughly the same time complexity estimate of [Kra02]. In contrast, we evaluated our attacks against the *real* E_0 cipher, and show a greatly improved and practical space complexity of 2^{23} BDD nodes (without the $O()$ notation).

Currently, the best *long keystream attack* against E_0 is by Y. Lu, W. Meier and S. Vaudenay in [LMV05]. The attack is a conditional correlation attack on the two-level Bluetooth E_0 , that fully recovers the original encryption key using the first 24 bits of $2^{23.8}$ frames with $O(2^{38})$ computations. Since it is against the two-level cipher, the attack is not limited to a single continuous Bluetooth frame—so the requirement of $2^{23.8}$ frames is attainable in principle.

Another BDD-based cryptanalysis attack against a different cryptosystem was presented by J.F Michon, P. Valarcher and J.B Yunés in [MVY03]. They used BDD's to implement a ciphertext only attack against HFE (Hidden Field Equations - a public key cryptosystem). They report that the attack was not efficient.

1.3 Contributions

In this paper we describe an implementation of an attack against E_0 that is based on the use of *Binary Decision Diagrams (BDD's)*. Our attack is based upon the theoretical BDD-based attack framework of M. Krause [Kra02]. Krause's work covered several keystream generators including the E_0 ; Consequently, we needed to supply missing details to adjust the attack for the E_0 system. Furthermore, we discovered that Krause's general attack can be greatly simplified and optimized when it is used against E_0 : We discovered that it is possible to use OBDD's rather than FBDD's throughout the algorithm; We re-engineered the algorithm to adjust to the different LFSR lengths; We developed an efficient composable BDD for the *Blender*; and after discovering that standard BDD algorithms and libraries are very inefficient for this algorithm we wrote our own BDD code that is optimized for attacking E_0 .

In addition, we built several hybrid variants of the basic BDD-based algorithm. These variants include: (i) partially guessing LFSR's initial data, (ii) using an intentionally "defective" Blender, and (iii) enumerating the satisfying assignments and testing them. We evaluated our attacks against the full, non-reduced, E_0 cipher. Our best heuristics can recover the initial state of the LFSR's, for the first time, with a practical space complexity of 2^{23} (84MB RAM). Our time complexity is 2^{87} : slightly higher complexity than reported by [Kra02], [KS06]—however, the attack is massively parallelizable. In addition to the specifics of Bluetooth, our work describes practical experience with BDD-based cryptanalysis, which so far has mostly been a theoretical concept.

Organization: In Section 2 we give an overview of the E_0 cipher, a brief overview of Binary Decision Diagrams and a description of Krause's attack. Section 3 describes adapting the attack to E_0 and analyzes the theoretical complexity of the attack. Section 4 describes the implementation of the attack, the heuristics used to lower attack complexity, and the performance we achieved. Section 5 concludes our work. Appendix A contains a detailed explanation of the bounds used in the theoretical complexity analysis of Section 3.4.

2 Preliminaries

2.1 Overview of the E_0 System

A full specification of Bluetooth security mechanisms can be found in part H of Vol 2 of [Blu03]. The security layer of Bluetooth, which is a part of the link layer, includes key management and key generation mechanisms, a challenge-response authentication scheme, and a data encryption engine. The data encryption engine used within Bluetooth is the E_0 keystream generator.

Table 1. The finite state machine transition function. NS stands for *Next State*. Each of the five main columns stands for a possible sum of the 4 LFSR bits that is input to the state machine.

Current State	Input									
	0		1		2		3		4	
	Out	NS	Out	NS	Out	NS	Out	NS	Out	NS
0	0	0	1	0	0	4	1	4	0	8
1	0	12	1	12	0	8	1	8	0	4
2	0	4	1	4	0	0	1	0	0	12
3	0	8	1	8	0	12	1	12	0	0
4	1	5	0	1	1	1	0	13	1	13
5	1	9	0	13	1	13	0	1	1	1
6	1	1	0	5	1	5	0	9	1	9
7	1	13	0	9	1	9	0	5	1	5
8	0	14	1	14	0	2	1	2	0	6
9	0	2	1	2	0	14	1	14	0	10
10	0	10	1	10	0	6	1	6	0	2
11	0	6	1	6	0	10	1	10	0	14
12	1	11	0	7	1	7	0	3	1	3
13	1	7	0	11	1	11	0	15	1	15
14	1	15	0	3	1	3	0	7	1	7
15	1	3	0	15	1	15	0	11	1	11

E_0 is initialized using a 128 bit session key (denoted K'_c), the Bluetooth address of the master device and a clock, which is different for every packet. Details regarding the generation of K'_c appear in section 2.2. E_0 generates a binary keystream, K_{cipher} , which is xor-ed with the plaintext. The cipher is symmetric; decryption is performed in exactly the same way using the same key as used for encryption.

The E_0 system employs four *linear shift feedback registers* (LFSR's), of lengths 25, 31, 33, and 39 (total length of 128 bits), a *Summation Combiner Logic* and a non-linear *Blend machine*. We can represent the *summation combiner logic* and the *blend machine* together as a 4 bit finite state-machine. At each clock tick the LFSR's are clocked once, and the output of the four LFSR's is xor-ed with the output bit of the finite state machine, to create the next output bit of the encryption stream K_{cipher} . The sum of the four output bits of the LFSR's is input into the finite state machine to update the state of the machine. In the remainder of this paper, the finite state machine will be denoted as the ***Blender*** unit. The finite state machine transition function (following [LV04], [LW05]) can be found in Table 1.

2.2 Detailed Specifications of the Encryption System

When two Bluetooth devices wish to establish a secure communication link, they first undergo through the pairing and authentication process. The specific details of this process are not given in this paper, see [SW05] for the full details

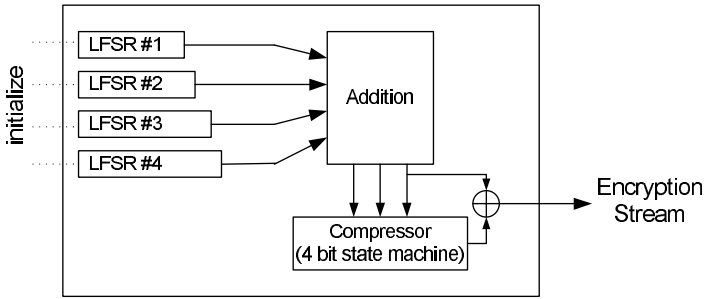


Fig. 1. The E_0 System

of this process. At the end of this process, both devices hold a 128 bit secret key (the link key, K_{ab}). This key is stored in a non-volatile memory area of the two devices, for future communication between these devices. This key is used to generate the *encryption key* (K_c), also known as the *session key*. Using an algorithm (E_3), both devices derive the encryption key from the link key (K_{ab}), a ciphering offset number (COF), that is generated during the authentication process done prior to the encryption phase, and a public known random number (EN_RAND) that is exchanged between the devices. The encryption key (K_c) is then modified into another key denoted K'_c . This modification is done to lower the effective size of the session key, according to the effective length the devices have decided upon negotiation in a preliminary phase. K'_c is used in a linear manner, along with some publicly known values (the Bluetooth address of the master device and a clock, which is different for every packet) to form the initial value of E_0 , for a two level keystream generator. E_0 generates a binary keystream, K_{cipher} , which is xor-ed with the plaintext. The cipher is symmetric; decryption shall be performed in exactly the same way using the same key as used for encryption.

2.3 Binary Decision Diagrams

A binary decision diagram (BDD) is a data structure that is used to represent a Boolean function. Let \mathbf{X}_n denote the set of boolean variables (x_0, \dots, x_{n-1}) of some boolean function. A **BDD** P over X_n is a rooted, directed, acyclic graph where each non-terminating node is labeled by a query $(x_i?)$ and has outdegree two, one edge labeled 0 and one edge labeled 1, connecting to child nodes. There are two terminating nodes: one 0-sink and one 1-sink. The root node is considered the source node. Each assignment $w(x_0 = w_0, x_1 = w_1, \dots, x_{n-1} = w_{n-1})$ where $w_i \in \{0, 1\}$ defines a unique path in P , which starts at the source node, answers w_i on queries $(x_i?)$ and always leads to a unique sink. The ending sink is the result of the boolean function under the assignment w . Two BDD's are considered equivalent if they compute the same boolean function.

A BDD is a **Free Binary Decision Diagram (FBDD)** if along each path in the BDD each variable appears at most once.

A BDD is an **Ordered Binary Decision Diagram (OBDD)** if on **all** paths in the BDD the variables respect a given ordering $x_0 < x_1 < x_2 < \dots < x_{n-1}$ (While FBDD's allow different orderings along each path).

2.4 BDD-Based Cryptanalysis of E_0

Problem model: The general attack framework of Krause [Kra02] works as follows. Given some known keystream bits, we would like to calculate the initial value of the LFSR's. Let $L(x)$ denote the internal linear bitstream in the E_0 keystream generator. $L(x)$ is actually comprised of the output sequence of the *four parallel LFSR's* in E_0 . E.g., for an E_0 keystream of 128 bits, $L(x)$ comprises of 512 bits. Let $C(z)$ denote the non-linear component in E_0 . $C(z)$ is actually the *Blender* unit, including the output xor operation that is used to derive the keystream. According to these declarations, K_{cipher} equals $C(L(x))$, where x is the secret initial value of the LFSR's.

Krause's observation is that finding a secret key x fulfilling $K_{cipher} = C(L(x))$ for a given keystream K_{cipher} , is equivalent to the problem of finding the minimal FBDD P for the decision whether x fulfills $K_{cipher} = C(L(x))$. This idea is the basis for the BDD attack against the E_0 system.

The algorithm: Let $L(x)$, $C(z)$ and K_{cipher} be as before. Let n be the key length (=128).

1. For all $m \geq 1$ let Q_m denote a minimal FBDD which decides for $z \in \{0, 1\}^m$ whether $C(z)$ is a prefix of K_{cipher} . In other words, Q_m is a FBDD which is built based on the value of the known keystream bits (K_{cipher}). This FBDD receives prefixes of the internal bitstreams which are generated by each LFSR as input. If this internal bitstream generates a prefix of the known keystream bits (K_{cipher}) - the FBDD accepts it. Otherwise, the FBDD rejects the input.
2. For all $m \geq n$ let S_m denote a minimal FBDD which decides for $z = (z_0, z_1, \dots, z_m)$ whether $z_m = L(z_0, z_1, \dots, z_{n-1})$. In other words, S_m is a FBDD which is built based on the feedback polynomials of the LFSR's. This FBDD receives the initial value of the LFSR's as input. If this initial value generates the correct value of z_m (the m -th internal stream bit) - the FBDD accepts it. Otherwise, the FBDD rejects the input.
3. Construct a third set of FBDD's, denoted P_m , which is the minimal FBDD which decides whether $z \in \{0, 1\}^m$ is a linear bitstream generated via L **and** if $C(z)$ is a prefix of K_{cipher} . Note that P_m is actually the result of the intersection between Q_m and S_m : $P_m = SYNTH(Q_m, S_m)$ — where $SYNTH$ denotes the BDD synthesis operation (cf. [Weg00]).

The strategy of Krause's algorithm is as follows: It incrementally computes P_m for increasing values of m until only one assignment will be accepted by P_m . This assignment is the initial value of the LFSR's generating K_{cipher} .

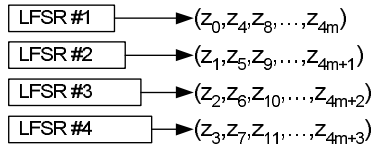


Fig. 2. Indexing method used in implementation

3 Adapting the Attack to E_0

3.1 Reduction of the Algorithm

The algorithm described by Krause is generic, and needs to be adapted for use on E_0 . We made the following reductions and changes before implementing the algorithm:

1. A key observation is that E_0 is regularly clocked. Every clock tick, one bit from each LFSR is input to the Blender, and each LFSR is stepped once. This regularity gives us two important advantages: First, E_0 induces a natural order on the internal bit stream Z : In our implementation, the variable ordering we used is: $\pi = (z_0, z_1, z_2, z_3, z_4, \dots, z_j, \dots, z_{511})$: for $j = 4 * m + L_i - 1$ we have that m is the clock tick index ($0 \leq m \leq 127$), and L_i is the index of the LFSR ($1 \leq L_i \leq 4$). Figure 2 describes the indexing method we used in implementation of the algorithm. Second, we can switch from using FBDD's to using OBDD's. This switch can be done, since on each path of the BDD we are creating, the variable ordering is the same. This has critical implementation benefits, since the data structures for supporting OBDD's are much simpler and more efficient than those of FBDD's.
2. We needed to adjust for the fact that the four LFSR's in E_0 have different lengths. This changes the implementation details and the complexity analysis.
3. As Section 2.4 implies, we had to implement a synthesis operation between two BDD's. Our implementation was based on the synthesis algorithm suggested by Wegener (See Section 3.3 of [Weg00]). However, we found that (1) all our BDD's are OBDD's; (2) none of them contain a self loop; and (3) all our BDD's are already reduced (minimal in size); Therefore, the use of a hash table in the algorithm is redundant and can be eliminated. This modification made our code specific for the E_0 attack—but it tremendously improved the performance of the algorithm in comparison with general purpose BDD libraries that we tried to use.

3.2 Building the *LFSR* Consistency OBDD

As described in Section 2.4, S_i denotes the BDD that computes whether the internal bit z_i is consistent with the prefix $\{z_j\}_{j=1}^{i-1}$. Since each internal bit is produced by one of the LFSR's, its consistency depends on 4 earlier bits of the

Table 2. LFSR’s consistency equations

LFSR #	Basic consistency equation	Normalized consistency equation
1	$z_i = z_{i-8} \oplus z_{i-12} \oplus z_{i-20} \oplus z_{i-25}$	$z_i = z_{i-32} \oplus z_{i-48} \oplus z_{i-80} \oplus z_{i-100}$
2	$z_i = z_{i-12} \oplus z_{i-16} \oplus z_{i-24} \oplus z_{i-31}$	$z_i = z_{i-48} \oplus z_{i-64} \oplus z_{i-96} \oplus z_{i-124}$
3	$z_i = z_{i-4} \oplus z_{i-24} \oplus z_{i-28} \oplus z_{i-33}$	$z_i = z_{i-16} \oplus z_{i-96} \oplus z_{i-112} \oplus z_{i-132}$
4	$z_i = z_{i-4} \oplus z_{i-28} \oplus z_{i-36} \oplus z_{i-39}$	$z_i = z_{i-16} \oplus z_{i-112} \oplus z_{i-144} \oplus z_{i-156}$

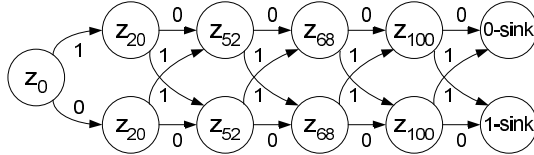


Fig. 3. Example of an OBDD representing the LFSR-1 consistency check for bit Z_{100}

same LFSR as determined by the LFSR’s taps. For example, for the shortest LFSR each bit must comply with the LFSR feedback polynomial: $t^{25} + t^{20} + t^{12} + t^8 + t^0$; meaning, bit z_i equals :

$$z_i = z_{i-8} \oplus z_{i-12} \oplus z_{i-20} \oplus z_{i-25} \tag{1}$$

Using our bit ordering (see Figure 2) changes the equation to:

$$z_i = z_{i-32} \oplus z_{i-48} \oplus z_{i-80} \oplus z_{i-100} \tag{2}$$

Table 2 summarizes the basic consistency equations and the normalized consistency equations for all four LFSR’s. Note that $LFSR_i$ produces bits with index j such that $j \equiv (i - 1) \pmod 4$.

Notation: For register L_i of length $|L_i|$, we call the first $|L_i|$ bits in its bit stream (bits $\{Z_k\} : k = 4j + L_i - 1 \text{ for } 0 \leq j \leq |L_i| - 1$) its *native bits*. The goal of the algorithm is to compute the native bits of all 4 LFSR’s (128 bits in total).

An OBDD representing an LFSR consistency condition contains 5 variables and 11 nodes (including the 0 sink and 1 sink). Figure 3 shows the OBDD which checks the consistency condition for bit number 100. Note that a different number of OBDD’s is created for each LFSR; this is because each LFSR is of different length and produces a different number of *non-native* bits. The number of *non-native* bits each LFSR produces equals to the keystream length minus the size of the LFSR. Therefore, the total number of OBDD’s representing an LFSR consistency condition is $4n - 128$ which is 384 (since $n = 128$).

3.3 Building the *Blender* OBDD

The OBDD representing the non-linear component of E_0 (denoted Q_m in Section 2.4) represents the *Blender* unit (see Section 2.1). This OBDD is built according to the known keystream bits, and according to the transition function of the Blender (see Table 1).

As stated before, the Blender updates its value according to the sum of the LFSR’s output bits. Therefore, we need a BDD structure to represent the sum

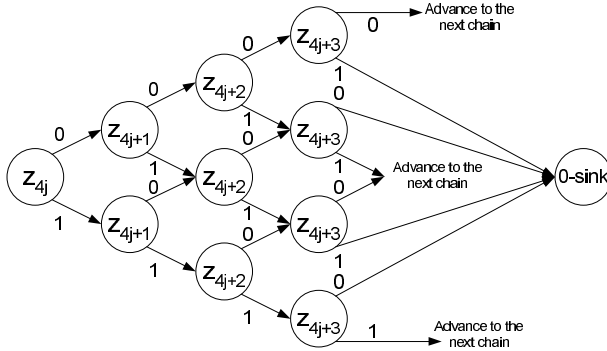


Fig. 4. The structure of a single basic chain in the Blender

of 4 bits. We call such a structure a *basic chain*. For each state and each of the 5 possible sums, Table 1 tells us what the output bit should be. If it matches the bit given in the known keystream, we can advance to the next chain, and test the next four bits; Otherwise, this path will lead to the 0-sink. Figure 4 shows the structure of a basic chain. Table 1 shows that for all states, exactly half the paths advance to the next chain, and the other half are connected directly to the 0-sink.

The Blender BDD is built from blocks, each consisting of 16 basic chains (one for each possible state of the Blender). Half the paths from each block lead to the 0-sink, while the other half advance to appropriate states on the next block. Figure 5 illustrates the full structure of the OBDD representing the Blender.

A single Blender block consists of 160 nodes and uses 4 (consecutively numbered) bits. Note, though, that each of the 4 bits “contributes” a different number of nodes to a block. Furthermore, attaching a sequence of blocks produces a non-minimal BDD, which can be reduced. For instance, for 128 blocks, the reduced Blender BDD consists of $\approx 14,500$ nodes, rather than 20,480.

3.4 Theoretical Complexity Analysis

The time complexity of the algorithm is determined by the space complexity of the synthesized OBDD throughout the entire process of synthesis. At any stage in the process, the size of the synthesized OBDD is bounded by two bounds (See Wegener [Weg00]):

1. The number of assignments satisfying the OBDD bounds the size of the minimal OBDD representing that boolean function:

$$|P| \leq m \cdot |One(P)| \quad (3)$$

where $One(P)$ denotes the set of satisfying assignments of the BDD P , and m is the number of variables the BDD contains ($m : 4 \rightarrow 512$).

2. Each synthesis operation bounds the size of the synthesis result: In general, the bound is $|SYNTH(P, Q)| \leq |P| \cdot |Q|$. However, when P is an LFSR consistency check OBDD, we can use a tighter bound. This is mainly due

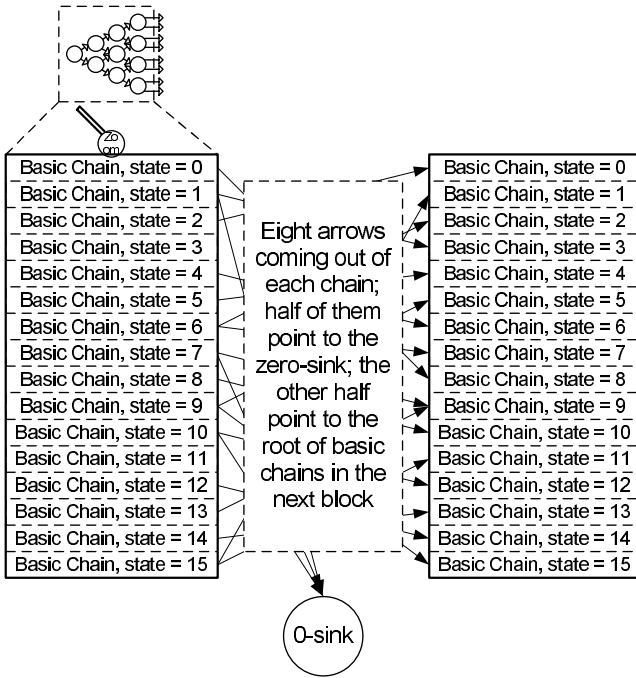


Fig. 5. Two consecutive blocks in an OBDD representing the Blender

to the structure of the OBDD’s representing the LFSR’s consistency check; These OBDD’s effectively keep a parity bit to “remember” if the consistency is held at each point. This is why each variable appears twice in the LFSR consistency OBDD. When synthesizing another OBDD with an LFSR consistency OBDD, each node within the “window” of the parity between the lowest and highest numbered variables in the LFSR consistency OBDD is duplicated, therefore the resulting OBDD must be at most twice the size of the larger OBDD. This bound can be summed in:

$$|P| \leq |Q(m)| \cdot 2^{m-n} \tag{4}$$

where $|Q(m)|$ is size of the OBDD representing the Blender, m is the number of variables ($m : 4 \rightarrow 512$) and n is the amount of given keystream ($n : 1 \rightarrow 128$ bits). Note that this bound is still loose because only nodes within the “window” of the parity are duplicated, while this bound assumes that all OBDD nodes are duplicated.

The bound on the size of the OBDD throughout the process is the lower envelope of bounds (3) and (4). Figure 6 shows the two bounds.

Using (3) (number of satisfying assignments), we get that during the first steps, each clock tick introduces 4 new variables, and one constraint since the output bit is known. This means the number of satisfying assignment is multiplied by 2^3 in each clock tick. Once we pass 25 clock ticks, all the native bits of

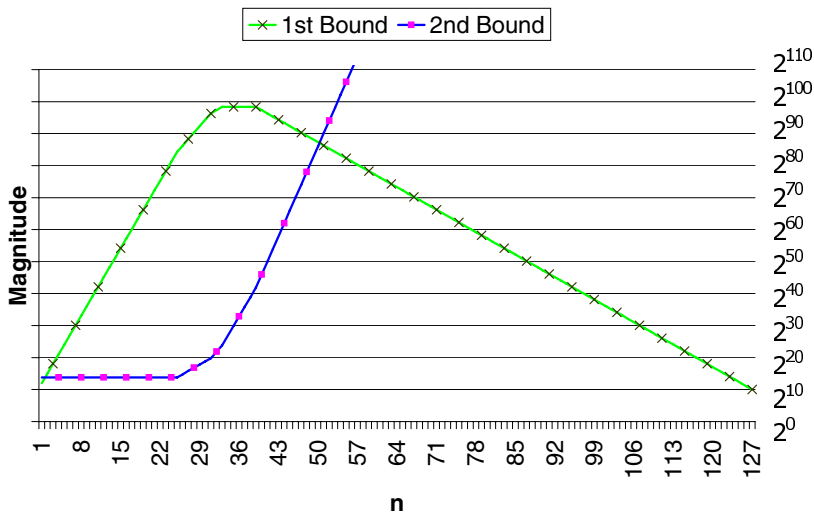


Fig. 6. The two bounds

LFSR #1 are fully determined, so the number of satisfying assumptions grows by a factor of 2^2 per clock tick. When the native bits of all four LFSR's are already set due to the consistency condition of the LFSR's (i.e., when $n \geq 39$), the number of satisfying assignments starts to decrease by half on each clock tick. The bound due to the number of satisfying assignment for $n \geq 39$ is $|P| \leq m \cdot 2^{128-n}$. See appendix A.1 for a detailed calculation of this bound.

Using (4) (magnitude of the synthesis result), we get that as long as we didn't start synthesizing with LFSR consistency OBDD's ($n \leq 25$), the OBDD size is at most the size of the OBDD representing the Blender (C_OBDD). When we begin the synthesis operation, the OBDD starts growing by a factor of 2 for each synthesis operation. Note that the number of synthesis operations for one tick depends on n . The bound due to the magnitude of the synthesis result for $n \geq 39$ is $|P| \leq |C_OBDD| \cdot 2^{4n-128}$ (The size of C_OBDD is approximately 2^{14}). See appendix A.2 for a detailed calculation of this bound.

Calculating the intersection point of the two bounds, we get that the maximal size of the OBDD synthesized throughout the process is $|P| \approx 2^{86}$. This maximal size of the OBDD appears at clock tick $n = 50$. This gives a total time complexity of $O(2^{90})$, since we need to run the algorithm with different value of the 4 bits initializing the state machine. Note that this estimate is significantly larger than the quick estimation made by Krause. However this is still a relatively **loose** bound; The actual size of the OBDD synthesized throughout this process is in fact lower. To refine this bound, we ran a simulation which builds a histogram representing the number of nodes in the synthesized OBDD for each bit index. Using the simulation results we calculated that the maximal size of the OBDD synthesized during the process is $|P| \approx 2^{82.5}$. This gives a total time complexity of $O(2^{86.5})$, i.e., the BDD attack is roughly equivalent to the attack of [FL01] in terms of time complexity.

Table 3. Complexity results for different numbers of guessed bits

Total number of guessed bits in LFSR's #3+#4	Maximal OBDD size (# nodes)	Total time complexity
12	$2^{18.3}$	$2^{90.3}$
10	$2^{18.7}$	$2^{88.7}$
8	$2^{19.9}$	$2^{87.9}$
6	$2^{21.7}$	$2^{87.7}$
4	$2^{23.4}$	$2^{87.4}$

4 Advanced Heuristics

Since running the algorithm as-is would take impractically long, and would require an unreasonable amount of memory, we used several heuristics to lower the time and space complexity of our attack.

4.1 Guessing Initial LFSR Bits

The first idea was to **guess** the value of some initial LFSR bits and use the BDD method only in the remaining bits. This gives us two advantages: (a) Lower space complexity, since the size of the OBDD representing the Blender is lower, and more importantly the number of OBDD's one has to synthesize with is significantly lower. (b) This idea also allows parallelization of the attack, since one can run the algorithm with different values of guessed bits on different machines.

On our test computer (a Pentium IV with 1Gb RAM running WinXP) we were only able to run the BDD attack by guessing all 56 bits of LFSR's #1 and #2, plus a few bits of LFSR's #3 or #4 (or both). When we guessed fewer bits, the program exhausted all the available RAM and failed to complete. The best results were obtained when guessing the entire content of LFSR's #1 and #2 plus another four bits, two bits from each of the remaining LFSR's. The latter were located at the end of LFSR's #3 and #4. In this case the maximal size of the OBDD synthesized was $\approx 2^{23}$ nodes, which used 84Mb RAM¹; Since we guess a total number of 60 bits (25+31+4), and we have to run the algorithm for all possible initial states of the Blender (4 bits), the total time complexity is $O(2^{87})$. Table 3 summarizes the results obtained when trying to run the algorithm with different numbers of guessed bits in LFSR #3 and #4.

4.2 Changing the Position of the Guessed Bits

Another heuristics we tested was to change the position of the 4 guessed bits in LFSR #3 and LFSR #4. Recall that these guessed bits were originally selected

¹ The program needs to maintain two such data structures during the synthesis operation, plus various other data structures. We observed that the program's peak RAM usage reached about 400MB.

at the end of the two LFSR's, so we decided to test how changing their location would affect the attack's complexity. The positions we tried include:

1. Guessing 2 *native* bits at the end of each LFSR (original position).
2. Guessing *native* bits that are positioned exactly where the LFSR taps are.
3. Guessing the first *non-native* bits of each LFSR.
4. Guessing bits only from one LFSR (#3 or #4).
5. Guessing bits from parallel positions in LFSR #3 and #4.

The reason for trying to guess bits on the LFSR taps positions (test #2) is that this can cause a single LFSR consistency OBDD (See Section 3) that is used during the synthesis procedure, to be totally eliminated.

However, the best results were obtained when the guessed bits were located at the end of the LFSR's (i.e., in the original bit positions). All the other alternatives increased the maximal OBDD size by factors of 2–4. Thus, the time complexity in this case is $O(2^{87})$ and the space complexity is $O(2^{23})$.

4.3 Using an Intentionally Defective Blender

A close examination of the transition function of the Blender (see Table 1) shows that from every state there are only 3 possible next states. Furthermore, the probability of entering each of these states is not uniform; For every state, there exists one next state that is reached with probability 1/16. For example, if we look at the reachable states from state #0, we note that state #8 is reachable with probability of 1/16. This leads to our next suggested heuristic: build a Blender that lacks the low-probability transition in every state. Naturally, this causes our attack to fail, if one bit of the known keystream was generated using such a transition. Therefore, instead of eliminating all the low probability transitions, we eliminate them only on the first 32 blocks of the Blender BDD. This means that the probability of performing a successful attack on a given known keystream is $(15/16)^{32} = 12.6\%$. This heuristic lowered the size of the synthesized OBDD by 14%. Thus, the overall complexity of the attack using an intentionally defective Blender has decreased, but is still around $O(2^{87})$.

4.4 Changing the Order of Synthesis

Another type of heuristic we tried was to change the order in which the OBDD's are synthesized: the order in which the various LFSR consistency OBDD's are synthesized does not affect the final outcome. The default synthesis order was by increasing bit index order. However, we conjectured that the OBDD will grow more slowly if we order the synthesis so all the LFSR OBDDs that "hit" some Blender block are synthesized consecutively, then those that hit some other Blender block, etc. We built a simulation to calculate the best order using the above criterion, and then ran the algorithm using the order produced by the simulation. Unfortunately, this heuristic produced poor results: the attack in which 4 bits of LFSR's #3 and #4 are guessed crashed for lack of memory (whereas the same attack using the default order ran to completion).

4.5 Enumerating Satisfying Assignments

The typical failure mode of the BDD attack is that all available memory is exhausted. However, just before such a failure occurs, we can trade time for the missing space, and still run the attack to completion. The idea is to stop the synthesis operation when the synthesized OBDD is close to the memory upper limit. Then, we enumerate all the satisfying assignments for the last synthesized OBDD, and test each assignment by generating the corresponding keystream for that assignment and comparing it to the given keystream. The overall complexity of this procedure is dominated by either the size of the synthesized OBDD or the number of satisfying assignments, whichever is larger. The time complexity of this approach is obviously poorer than using the previous heuristics—it's main advantage is that it allows one to obtain results even if the available RAM is insufficient.

5 Conclusion

We have presented an implementation of a BDD-based attack that is a short keystream cryptanalysis of the E_0 cipher. We have shown that several significant reductions and changes needed to be made to Krause's general attack. These changes include using OBDD's instead of FBDD's, using the exact size of the LFSR's, and skipping the use of a hash table in the implementation of the synthesis operation. We also performed an accurate complexity analysis of this attack. Furthermore, we presented some heuristics that lower the time and space complexity of this attack, and to allow parallelization of the attack on multiple machines. Our best heuristic has a time complexity which is roughly equivalent to that of the attacks of S. Fluhrer and S. Lucks [FL01] and O. Levy and A. Wool [LW05], and has significantly better space complexity than the recent work of Krause and Stegemann [KS06].

References

- [Blu03] Specification of the Bluetooth system, v.1.2. Core specification, available from <http://www.bluetooth.org/spec>, 2003.
- [EJ00] Patrik Ekdahl and Thomas Johansson. Some results on correlation in the bluetooth stream cipher. In *Proc. of the 10th Joint Conference on Communication and Coding*, 2000. Obertauern, Austria, March 2000.
- [FL01] Scott R. Fluhrer and Stefan Lucks. Analysis of the E_0 encryption system. In *Proc. 8th Workshop on Selected Areas in Cryptography, LNCS 2259*. Springer-Verlag, 2001.
- [GBM02] Jovan Dj. Golic, Vittorio Bagini, and Guglielmo Morgari. Linear cryptanalysis of Bluetooth stream cipher. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*. Springer-Verlag, 2002.
- [HN99] Miia Hermelin and Kaisa Nyberg. Correlation properties of the Bluetooth combiner generator. In *Information Security and Cryptology, LNCS 1787*, pages 17–29. Springer-Verlag, 1999.

- [JW01] Markus Jakobsson and Susanne Wetzel. Security weaknesses in Bluetooth. In *Proc. RSA Security Conf. – Cryptographer's Track, LNCS 2020*, pages 176–191. Springer-Verlag, 2001.
- [Kra02] Matthias Krause. BDD-based cryptanalysis of keystream generators. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT'02, LNCS 1462*, pages 222–237. Springer-Verlag, 2002.
- [KS06] Matthias Krause and Dirk Stegemann. Reducing the space complexity of BDD-based attacks on keystream generators. In *13th annual Fast Software Encryption Workshop (FSE 2006)*, Graz, Austria, March 2006.
- [LMV05] Y. Lu, W. Meier, and S. Vaudenay. The conditional correlation attack: A practical attack on Bluetooth encryption. In *Advances in Cryptology – CRYPTO'05, LNCS 3621*, pages 97–117. Springer-Verlag, 2005.
- [LV04] Y. Lu and S. Vaudenay. Faster correlation attack on Bluetooth keystream generator E_0 . In *Advances in Cryptology – CRYPTO'04, LNCS 3152*, pages 407–425. Springer-Verlag, 2004.
- [LW05] O. Levy and A. Wool. A uniform framework for cryptanalysis of the Bluetooth E_0 cipher. In *Proc. 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, pages 365–373, Athens, Greece, September 2005.
- [MVY03] Jean-François Michon, Pierre Valarcher, and Jean-Baptiste Yunés. HFE and BDDs: A practical attempt at cryptanalysis. In *International Workshop on Coding Cryptography and Combinatorics, Huangshan (China)*, June 2003.
- [Saa00] Markku-Juhani O. Saarinen. Re: Bluetooth und E_0 . Post to sci.crypt.research, September 2000.
- [SW05] Yaniv Shaked and Avishai Wool. Cracking the Bluetooth PIN. In *Proc. 3rd USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys)*, pages 39–50, June 2005.
- [Weg00] Ingo Wegener. *Branching programs and binary decision diagrams*. SIAM, 2000.

Appendix

A Detailed Bounds Calculation

A.1 Bound Due to the Number of Satisfying Assignments

Using the first bound term, we get that:

$n = 1$	$ P \leq m \cdot 2^3$	On the first step, we have 3 free bits, and the last bit is determined
$n = 2$	$ P \leq m \cdot 2^6$	Same for the next step
$n \leq 25$	$ P \leq m \cdot 2^{3n}$	Same for the next steps, as long as we take initial bits from LFSR #1
$25 \leq n \leq 31$	$ P \leq m \cdot 2^{75} \cdot 2^{2n-25}$	One bit is already set due to the consistency condition of LFSR #1; So we have two free bits, and the last bit is determined

$$31 \leq n \leq 33 \quad |P| \leq m \cdot 2^{75} \cdot 2^{12} \cdot 2^{n-31}$$

Two bits are already set due to the consistency condition of LFSR's #1-#2; So we have one free bit, and the last bit is determined

$$33 \leq n \leq 39 \quad |P| \leq m \cdot 2^{75} \cdot 2^{12} \cdot 2^2$$

Three bits are already set due to the consistency condition of LFSR's #1-#3; The last bit is determined

$$39 \leq n \quad |P| \leq m \cdot 2^{75} \cdot 2^{12} \cdot 2^2 \cdot 2^{39-n}$$

Four bits are already set due to the consistency condition of LFSR's #1-#4; Only half of the satisfying assignments survive in each step

A.2 Bound Due to Magnitude of the Synthesis Result

$$n \leq 25$$

$$|C_OBDD|$$

No synthesis operations done so far, since all bits are native

$$25 \leq n \leq 31$$

$$|C_OBDD| \cdot 2^{n-25}$$

One synthesis operation per each bit produced by LFSR #1

$$31 \leq n \leq 33$$

$$|C_OBDD| \cdot 2^{n-25} \cdot 2^{n-31}$$

Two synthesis operations per each tick; For the two bits produced by LFSR's #1-#2

$$33 \leq n \leq 39$$

$$|C_OBDD| \cdot 2^{n-25} \cdot 2^{n-31} \cdot 2^{n-33}$$

Three synthesis operations per each tick; For the three bits produced by LFSR's #1-#3

$$39 \leq n$$

$$|C_OBDD| \cdot 2^{n-25} \cdot 2^{n-31} \cdot 2^{n-33} \cdot 2^{n-39}$$

Four synthesis operations per each tick; For the four bits produced by LFSR's #1-#4

Where $|C_OBDD|$ denotes the size of the OBDD representing the compressor.

A Partial Key Exposure Attack on RSA Using a 2-Dimensional Lattice

Ellen Jochemsz* and Benne de Weger*

Department of Mathematics and Computer Science,
Eindhoven University of Technology, Eindhoven, The Netherlands
{e.jochemsz, b.m.m.d.weger}@tue.nl

Abstract. We describe an attack on the RSA cryptosystem when the private exponent d is chosen to be 'small', under the condition that a sufficient amount of bits of d is available to the attacker. The attack uses a 2-dimensional lattice and is therefore (in the area of the keyspace where it applies) more efficient than known attacks using Coppersmith techniques. Moreover, we show that the attacks of Wiener and Verheul/Van Tilborg, using continued fractions techniques, are special deterministic cases of our attack, which in general is heuristic.

Keywords: RSA, cryptanalysis, partial key exposure, lattice basis reduction, inhomogeneous diophantine approximation.

1 Introduction

Since the introduction of the RSA cryptosystem in 1977, people have been looking for its vulnerabilities. A summary of attacks on RSA up to 1999 was given by Boneh in [2]. Although none of these attacks totally break RSA, they provide a certain guideline for the use of RSA and show in which cases the cryptosystem is unsafe.

For instance, it is known that using a small private exponent d can be dangerous. In 1990, Wiener showed in [13] that if the size of d is less than $\frac{1}{4}$ th of the size of the modulus N , it can be found by continued fractions methods. Verheul and Van Tilborg [11] generalized this result in 1997, to obtain an attack based on continued fractions that works if d is slightly larger than $N^{\frac{1}{4}}$. In 2000, Boneh and Durfee [3] extended Wiener's bound to $d < N^{0.292}$.

The concept of partial key exposure attacks on RSA was introduced in 1997 by Boneh, Durfee and Frankel in [4], and deals with the situation where an attacker has obtained some bits of the private exponent d . The main question is: How much information on the bits of d is needed such that an attacker can reconstruct d , thereby breaking the RSA instance?

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The motivation for exploring partial key exposure attacks comes from side-channel attacks such as power analysis, timing attacks, etc. Using a side-channel, an attacker can expose a part of d , generally an MSB (most significant bit) part or LSB (least significant bit) part.

In all the subsequent papers about partial key exposure attacks, the assumption is made (besides knowledge of MSBs/LSBs of d) that one of the exponents e , d is chosen to be small (at least significantly smaller than the modulus N). This is a common practice, since a small exponent yields faster modular exponentiation. For instance, $e = 2^{16} + 1 = 65537$ is a popular choice, and for signing operations on constrained devices such as smartcards, it is useful to have the private (signing) exponent d to be small, though obviously larger than $N^{0.292}$.

The first partial key exposure attacks by Boneh, Durfee, and Frankel [4] required the public exponent e to be smaller than $N^{\frac{1}{2}}$. Blömer and May extended their result in [5] with attacks for $e \in [N^{0.5}, N^{0.725}]$. Ernst, Jochemsz, May, and De Weger [7] recently showed attacks for both the situations where the private exponent d or the public exponent e is chosen to be small. Both their attacks work up to full size exponents.

In the papers [3,5,7], lattice methods are used instead of continued fractions methods. Generally, one starts by describing an RSA situation in terms of an integer polynomial that has a small (unknown) root, or a polynomial that has a small (unknown) root modulo a known constant. After that, one uses the theory initiated by Coppersmith [6], to construct a lattice with polynomials with the same root, and reduce the lattice to obtain a polynomial, again having the same root, whose coefficients are small enough to find the root.

These attacks using lattice methods are asymptotic, meaning that if one comes close to the maximal value for the unknown part of d for which an attack should work, the lattices involved are very large. This implies that the lattice reduction phase, for which the LLL-algorithm [8] is used, may take a prohibitively long time.

Therefore, it may be useful to look at very small lattices instead of very large. In this paper, we explore for which sizes of d , one can mount an attack in a few seconds with a very simple method using a 2-dimensional lattice. Our result is summarized in the following theorem.

Theorem 1. *Under a reasonable heuristic assumption that we specify in Assumption 1, the following holds: Let $N = pq$ be an n -bit RSA-modulus, and p , q primes of bitsize $\frac{n}{2}$. Let $0 < \beta < \frac{1}{2}$, and let e , d satisfy $ed \equiv 1 \pmod{\phi(N)}$ with $\text{bitsize}(e) = n$ and $\text{bitsize}(d) = \beta n$. Given a (total) amount of $(2\beta - \frac{1}{2})n$ MSBs and/or LSBs of d (see Figure 1), N can be factored very efficiently, using a 2-dimensional lattice.*

We will comment on what 'very efficiently' means in Section 5, when we compare the performance of this attack on small d to the method of Ernst et al. [7]. Moreover, we show that the results of Wiener and Verheul/Van Tilborg can be obtained by our attack on small d and are simply special (homogeneous and deterministic) cases. One could also say that our partial key exposure attack

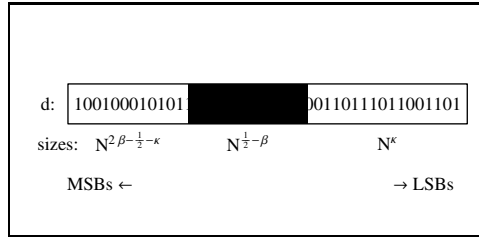


Fig. 1. Partition of d for small d

is the inhomogeneous counterpart of the results by Wiener and Verheul/Van Tilborg. We will comment on this, and on the heuristic assumption in the other cases, in Section 4.

The rest of this paper is organized as follows. In Section 2, we will state preliminaries on RSA and on lattice techniques, define some notation and introduce Assumption 1. Section 3 will contain the description of the attack for small d . In Section 4, we will comment on the cases where our attack does not depend on Assumption 1 and is therefore deterministic, and at the experimental results for the heuristic in the cases where we do need Assumption 1. In Section 5, we look at the efficiency of our method and compare our 2-dimensional attack with the existing partial key exposure attacks on small d of [7]. Finally, we will give a conclusion in Section 6.

2 Preliminaries on RSA and Lattices

In this section, we state some basic properties of RSA, the cryptosystem we are attacking and of 2-dimensional lattices, the tool we use to do so.

Let p, q, N, d, e be as usual, i.e. p and q are distinct primes, $N = pq$ is taken as modulus, and the encryption exponent e and decryption exponent d satisfy $ed \equiv 1 \pmod{\phi(N)}$. For the attack in this paper, we assume that p and q have the same bitsize, thus $p + q < 3N^{\frac{1}{2}}$. Let $k \in \mathbb{Z}$ be defined by the RSA key equation

$$ed - 1 = k\phi(N), \quad \text{where } \phi(N) = (p - 1)(q - 1) = N - (p + q - 1).$$

In our attack in this paper, we assume that the private exponent d is chosen to be small, for efficient modular computations. From the RSA key equation, it follows directly that $k < d$.

We define a 2-dimensional lattice L as the set of all integer linear combinations of two linearly independent vectors $\{\mathbf{b}_1, \mathbf{b}_2\}$, which are basis vectors. We usually say that L is the lattice spanned by the columns of the matrix $\Gamma = (\mathbf{b}_1 \ \mathbf{b}_2)$. The determinant of L is $\det(L) = |\det(\Gamma)|$, and though there are infinitely many bases possible, the determinant is always the same.

To find a small, so-called reduced basis $\{\mathbf{r}, \mathbf{s}\}$, one can use a reduction algorithm. For a 2-dimensional lattice, the Lagrange reduction algorithm (which

is simply a generalization of Euclid’s algorithm) finds a reduced basis, and this basis also contains the smallest nonzero vector of the lattice. We are interested in how small the reduced basis vectors are in norm.

We use the following notation for size-computations in this paper. With $u \approx N^\lambda$, we mean that u ‘has the size of’ N^λ , that is $|u| = C_u N^\lambda$ for some number C_u that does not deviate much from 1. Naturally, $\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \approx \begin{pmatrix} N^{\lambda_1} \\ N^{\lambda_2} \end{pmatrix}$ is a short notation for $v_1 \approx N^{\lambda_1}$ and $v_2 \approx N^{\lambda_2}$.

When we reduce the matrix Γ to $\Gamma_{\text{red}} = (\mathbf{r} \ \mathbf{s})$, with \mathbf{r} the smaller reduced basis vector and \mathbf{s} the larger reduced basis vector, it holds that $\|\mathbf{r}\| \cdot \|\mathbf{s}\| \approx \det(L)$. So, we assume $\|\mathbf{r}\| \approx a^{-1} \det(L)^{\frac{1}{2}}$ and $\|\mathbf{s}\| \approx a \det(L)^{\frac{1}{2}}$ for some $a \geq 1$.

Hence,

$$\Gamma_{\text{red}} = (\mathbf{r} \ \mathbf{s}) = \begin{pmatrix} r_1 & s_1 \\ r_2 & s_2 \end{pmatrix}, \text{ and } \Gamma_{\text{red}}^{-1} = \frac{1}{\det(\Gamma)} \begin{pmatrix} s_2 & -s_1 \\ -r_2 & r_1 \end{pmatrix} = \begin{pmatrix} \mathbf{s}'^{\mathbf{T}} \\ \mathbf{r}'^{\mathbf{T}} \end{pmatrix}.$$

It follows that the first row \mathbf{s}' of Γ_{red} satisfies $\|\mathbf{s}'\| \approx a \det(L)^{\frac{1}{2}}$. Analogously, $\|\mathbf{r}'\| \approx a^{-1} \det(L)^{\frac{1}{2}}$.

If the two reduced basis vectors \mathbf{r} , \mathbf{s} are ‘nearly-equal’ in length, that is when a does not deviate much from 1, then $\|\mathbf{r}\| \approx \|\mathbf{s}\| \approx \det(L)^{\frac{1}{2}}$. In other words, all reduced basis vectors of L have a norm of size $\det(L)^{\frac{1}{2}}$. However, it is also possible that there is one ‘extremely small’ basis vector, which makes the lattice ‘unbalanced’. For the attacks in this paper, we make the following assumption.

Assumption 1. *The reduced basis vectors given by the columns of Γ_{red} both have a norm of size $\det(L)^{\frac{1}{2}}$. In other words, the parameter a used to describe the unbalancedness of the lattice is near to 1.*

In Section 4, we comment on how this assumption holds in practice.

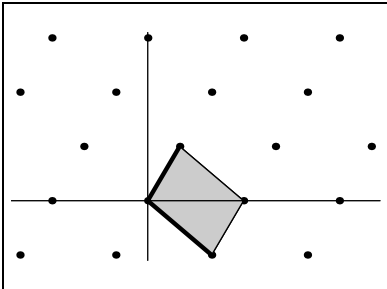


Fig. 2. $a \approx 1$

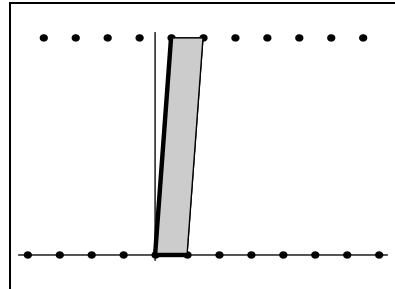


Fig. 3. $a \gg 1$

Having discussed the necessary preliminaries, we are now ready to explain the 2-dimensional partial key exposure attack on RSA for small d .

3 The Attack on Small d

3.1 Description of the Attack

Let $d = N^\beta < N^{\frac{1}{2}}$ and $e < \phi(N) < N$. In this section, we will prove the statement in Theorem 1, namely that we can factor N very efficiently if we know a (total) amount of $(2\beta - \frac{1}{2})n$ MSBs and/or LSBs of d .

This implies that our method will work if the 'unknown middle part' of d is of size N^δ with $\delta < \beta - (2\beta - \frac{1}{2}) = \frac{1}{2} - \beta$. The situation is sketched in Figure 4.

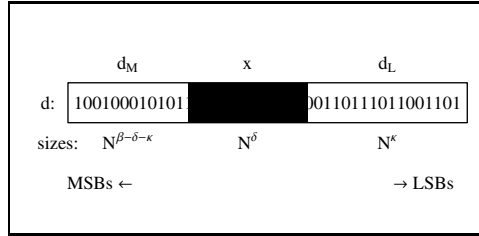


Fig. 4. Partition of d when MSBs and/or LSBs are known

Let d_L be the known LSB part of d of size N^κ , followed by an unknown middle part x of size N^δ , which itself is followed by a known MSB part d_M , of size $N^{\beta-\kappa-\delta}$. Hence, we can write

$$d = d_L + 2^{\lfloor \kappa n \rfloor} x + 2^{\lfloor \kappa n \rfloor + \lfloor \delta n \rfloor} d_M,$$

where $\lfloor \cdot \rfloor$ is simply rounding to the nearest integer.

When we substitute the partition of d in the RSA key equation, we obtain

$$e2^{\lfloor \kappa n \rfloor} x + ed_L + e2^{\lfloor \kappa n \rfloor + \lfloor \delta n \rfloor} d_M - 1 = k(N - (p + q - 1)).$$

Therefore, we must find the solution $(x, y, z) = (x, k, p + q - 1)$ of the trivariate equation

$$e2^{\lfloor \kappa n \rfloor} x - Ny + yz + R - 1 = 0, \text{ with } R = ed_L + e2^{\lfloor \kappa n \rfloor + \lfloor \delta n \rfloor} d_M.$$

The equation above implies that

$$|e2^{\lfloor \kappa n \rfloor} x - Ny + R| = |1 - yz| \leq |k(p + q - 1)| \leq |d(p + q)| \leq 3N^{\beta + \frac{1}{2}}.$$

This is an inhomogeneous diophantine approximation problem in the unknowns x and y . To solve it, we define a lattice L spanned by the columns of Γ , with

$$\Gamma = \begin{pmatrix} C & 0 \\ e2^{\lfloor \kappa n \rfloor} & N \end{pmatrix}, \text{ and } \mathbf{v} = \begin{pmatrix} 0 \\ -R \end{pmatrix},$$

where C is a convenient integer of size $N^{\beta-\delta+\frac{1}{2}}$.

The lattice point $\Gamma \begin{pmatrix} x \\ -y \end{pmatrix}$ is close to \mathbf{v} , since

$$\Gamma \begin{pmatrix} x \\ -y \end{pmatrix} - \mathbf{v} = \begin{pmatrix} Cx \\ e2^{\lfloor \kappa n \rfloor} x - Ny + R \end{pmatrix} \approx \begin{pmatrix} N^{\beta+\frac{1}{2}} \\ N^{\beta+\frac{1}{2}} \end{pmatrix}.$$

Our strategy to find x and y is therefore to start with a lattice vector \mathbf{v}' close to \mathbf{v} , and add small multiples of the reduced basis vectors of the lattice L until we get $\Gamma \begin{pmatrix} x \\ -y \end{pmatrix}$. To do so, we apply lattice basis reduction to the columns of Γ , and obtain a reduced matrix Γ_{red} , whose columns still span L . We aim to find an integer pair (z_1, z_2) for which

$$\Gamma_{\text{red}} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \Gamma \begin{pmatrix} x \\ -y \end{pmatrix} - \Gamma_{\text{red}} \lfloor \Gamma_{\text{red}}^{-1} \mathbf{v} \rfloor,$$

where $\lfloor \Gamma_{\text{red}}^{-1} \mathbf{v} \rfloor = \mathbf{v}'$ is the vector we get from rounding the elements of $\Gamma_{\text{red}}^{-1} \mathbf{v}$ to nearest integers. Alternatively, one could also solve the closest vector problem to obtain a lattice vector \mathbf{v}' to start with, but in practice the closest vector will almost immediately appear in this way as well.

It can be checked that

$$\Gamma_{\text{red}} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \left(\Gamma \begin{pmatrix} x \\ -y \end{pmatrix} - \mathbf{v} \right) - \left(\Gamma_{\text{red}} \lfloor \Gamma_{\text{red}}^{-1} \mathbf{v} \rfloor - \mathbf{v} \right) \approx \begin{pmatrix} N^{\beta+\frac{1}{2}} \\ N^{\beta+\frac{1}{2}} \end{pmatrix} + \Gamma_{\text{red}} \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix},$$

with $|\epsilon_i| < \frac{1}{2}$. Therefore

$$\begin{aligned} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} &\approx \Gamma_{\text{red}}^{-1} \begin{pmatrix} N^{\beta+\frac{1}{2}} \\ N^{\beta+\frac{1}{2}} \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix} = \begin{pmatrix} \mathbf{s}'\mathbf{T} \\ \mathbf{r}'\mathbf{T} \end{pmatrix} \begin{pmatrix} N^{\beta+\frac{1}{2}} \\ N^{\beta+\frac{1}{2}} \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix} \\ &\approx \begin{pmatrix} a \det(L)^{-\frac{1}{2}} N^{\beta+\frac{1}{2}} + \epsilon_1 \\ a^{-1} \det(L)^{-\frac{1}{2}} N^{\beta+\frac{1}{2}} + \epsilon_2 \end{pmatrix} \approx \begin{pmatrix} a N^{\frac{1}{2}(\beta+\delta-\frac{1}{2})} + \epsilon_1 \\ a^{-1} N^{\frac{1}{2}(\beta+\delta-\frac{1}{2})} + \epsilon_2 \end{pmatrix}. \end{aligned}$$

Each pair (z_1, z_2) leads to a pair $(x, -y)$. If we substitute x as the unknown part of d , and y as k , we can find a ϕ that satisfies $ed - 1 = k\phi$. First we test whether ϕ , computed as $\frac{ed-1}{k}$ is integral (unfortunately we see no way how to use this condition earlier). The next test will be to solve for the integer roots p, q of the quadratic equation $X^2 - (N + 1 - \phi)X + N = 0$.

The number of pairs (z_1, z_2) to try is of size

$$(aN^{\frac{1}{2}(\beta+\delta-\frac{1}{2})}) \cdot \max\{a^{-1}N^{\frac{1}{2}(\beta+\delta-\frac{1}{2})}, 1\}.$$

Hence, the number of pairs (z_1, z_2) to try is either

- $O(N^{\beta+\delta-\frac{1}{2}})$, when $a < N^{\frac{1}{2}(\beta+\delta-\frac{1}{2})}$, or
- $O(aN^{\frac{1}{2}(\beta+\delta-\frac{1}{2})})$, when $a > N^{\frac{1}{2}(\beta+\delta-\frac{1}{2})}$.

Note that in the latter case, $z_2 = 0$, but we do have to check for all z_1 separately.

In the next section, we show the relation between our method and the attacks of Wiener [13] and Verheul/Van Tilborg [11], which are special cases of this attack. For these situations, we show that the attacks are deterministic instead of heuristic, simply because the lattice vector $\Gamma \begin{pmatrix} x \\ -y \end{pmatrix}$ is small enough to ensure that the search region does not depend on a .

However, if we are outside the range of Wiener's and Verheul/Van Tilborg's attacks, it is highly unusual that the lattice involved contains an exceptionally small nonzero vector, which would make the lattice unbalanced and the attack inefficient. By Assumption 1, we take a to be close to 1. Under this heuristic, the number of pairs (z_1, z_2) to try is $O(N^{\beta+\delta-\frac{1}{2}})$. In Section 4.2, we will show that this assumption is reasonable in practice.

Under our heuristic assumption, and provided that δ is smaller than or at most only marginally larger than $\frac{1}{2} - \beta$, then we can efficiently try all pairs (z_1, z_2) and find the factorization of N .

One might note that by knowing MSBs of d , one can also obtain an MSB part of k . However, splitting k into a known and an unknown part results in more combinations of variables, which we can only represent in a 3-dimensional lattice instead of a 2-dimensional one. The 3-dimensional lattice attack will give a worse analysis than the method described in this section. This is an example of a common phenomenon in lattice based cryptanalysis, namely that sometimes one can get better results by leaving out information that one knows, just by the monomials of the equation involved.

3.2 Complexity

We now study the total complexity of the above attack.

Firstly it requires one lattice basis reduction for a 2-dimensional lattice. This is just Lagrange reduction, which takes at most $O((\log N)^3)$ bit operations.

Secondly, a number of $O(N^{\beta+\delta-\frac{1}{2}})$ pairs (z_1, z_2) have to be checked for coming from a solution. For each vector this check takes $O((\log N)^2)$ bit operations.

It follows that the bit complexity of our attack is $O((\log N)^3)$ when $\delta \leq \frac{1}{2} - \beta$, which is polynomial. When $\delta = \frac{1}{2} - \beta + \epsilon$ the bit complexity becomes exponential, namely $O(N^\epsilon(\log N)^2)$. This results in an increased workload by a factor N^ϵ . In other words, for an additional amount of r unknown bits, the complexity is equivalent to an exhaustive search over r bits. Furthermore, in the case that we let both d and the unknown part of d grow r bits, such that the known part of d stays of the same size, one can check that the extra workload will be an exhaustive search over $2r$ bits. This relates directly to a result of Verheul and Van Tilborg [11], on which we shall comment in Section 4.1.

3.3 Examples

We have done several experiments for this attack. A typical case is with 2048 bit N and $\delta = 0.156$, $\beta = 0.350$ (e.g. $\epsilon = 0.006$), meaning that d has about 717 bits, of which at most the 320 least significant bits are unknown.

Then $N^{\frac{1}{2}(\delta+\beta-\frac{1}{2})} \approx 70$. Indeed, we typically find a hit with $\|z\| \lesssim 200$. A search area like this takes only a few seconds with Mathematica 5 on a 2GHz Pentium 4 PC. And with $\delta \leq \frac{1}{2} - \beta$ typically $\|z\| \approx 1$, and the computation time is only a fraction of a second.

Here's a baby example for $\{\delta = 0.156, \beta = 0.35\}$. Let the 128-bit public key be given by

$$\begin{aligned} N &= 269866491905568049204176579604167754067, \\ e &= 222981052634419442506270512141611354797. \end{aligned}$$

Now suppose we know some MSBs of d , hence we know an approximation

$$\tilde{d} = 24584250313023$$

of d for which $d_0 = d - \tilde{d}$ is $0.156 \cdot 128 \approx 20$ bits. We take

$$\begin{aligned} C &= 2^{\lceil 128 \cdot (0.35 - 0.156 + 0.5) \rceil} = 2^{89}, \text{ and} \\ R &= e\tilde{d} = 5481822013025924218218657989757723471271758362621331, \end{aligned}$$

and we know that we are looking for $\{d_0, k\}$ such that

$$\Gamma \cdot \begin{pmatrix} d_0 \\ -k \end{pmatrix} - \mathbf{v} = \begin{pmatrix} C & 0 \\ e & N \end{pmatrix} \cdot \begin{pmatrix} d_0 \\ -k \end{pmatrix} - \begin{pmatrix} 0 \\ -R \end{pmatrix}$$

is a small vector. Then Γ_{red} is given by

$$\begin{pmatrix} 93923748720621086836871453999104 & -645630915298759729739927100850176 \\ 223858603616044679201441362439981 & 239654325473299927083414831489037 \end{pmatrix}$$

and $[\Gamma_{\text{red}}^{-1}v] = \begin{pmatrix} -21188034626414783992 \\ -3082348742879388262 \end{pmatrix}$.

We then enumerate the pairs $\{z_1, z_2\}$, for each value computing

$$\begin{pmatrix} x \\ -y \end{pmatrix} = \Gamma^{-1} \left(\Gamma_{\text{red}} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \Gamma_{\text{red}}[\Gamma_{\text{red}}^{-1}v] \right).$$

We try $d = \tilde{d} + x$ and $k = y$, and solve $N + 1 - \left(p + \frac{N}{p}\right) = \frac{ed - 1}{k}$ to get a possible factor p .

At $z = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$ we have a hit, namely $x = 1016998$, $y = 20313089635876$, so we find that $d = 24584251330021$, and $k = 20313089635876$.

It follows that $\phi(N) = 269866491905568049171299025219693706736$, and then we obtain the factors

$$\begin{aligned} p &= 15833051453602685849, \\ q &= 17044502930871361483. \end{aligned}$$

4 The Deterministic and Heuristic Cases of the Attack

4.1 Wiener and Verheul/Van Tilborg

In [11,13], attacks were described for small d . Wiener showed that when $d < N^{\frac{1}{4}}$, it can be found in polynomial time. Verheul and Van Tilborg's extension of Wiener's result shows the price for d slightly larger than this. Their attacks can be seen as homogeneous diophantine approximation problems, and continued fraction techniques are used to solve them.

In this section, we will show that Wiener's and Verheul/Van Tilborg's attacks are special cases of our method. Moreover, we will show that in these cases the method is deterministic, in other words, it does not depend on the size of a (the parameter that describes the unbalancedness of the lattice).

Wiener [13] bases his attack on the fact that $\frac{k}{d}$ can be found as a convergent of $\frac{e}{N}$ if

$$\left| \frac{e}{N} - \frac{k}{d} \right| < \frac{1}{2d^2}.$$

It is commonly known (see for instance [9]) that this can also be described using a 2-dimensional lattice. When we assume no part of d is known ($d_M = d_L = 0$), it follows that $R = 0$ and

$$\Gamma = \begin{pmatrix} C & 0 \\ e & N \end{pmatrix}, \quad \mathbf{v} = \mathbf{0},$$

with C of size $N^{\beta-\delta+\frac{1}{2}} = N^{\frac{1}{2}}$, will reproduce Wiener's result, namely that the method will work if $\beta < \frac{1}{4}$. Later in this section we will show that the solution will be found by the shortest lattice vector only, making this case deterministic.

Verheul and Van Tilborg [11] have given an extension of Wiener's attack, where d is at most slightly larger than $N^{\frac{1}{4}}$ and no bits are known. To find $\frac{k}{d}$, they look not only at convergents of $\frac{e}{N}$, but also at 'linear combinations' of consecutive convergents, which, be it not the best, nevertheless are pretty good approximations. To be precise, when $\frac{p_{i-1}}{q_{i-1}}, \frac{p_i}{q_i}$ are consecutive convergents, then they also look for approximations to $\frac{e}{N}$ of the form $\frac{\lambda p_i + \mu p_{i-1}}{\lambda q_i + \mu q_{i-1}}$ for parameters $\lambda, \mu \in \mathbb{N}$. Then they have a weaker inequality to satisfy, of the form of

$$\left| \frac{e}{N} - \frac{k}{d} \right| < \frac{c}{d^2},$$

where the exact value for c depends on the search region for λ and μ . In this way they show that in order to extend Wiener's result for $d < N^{\frac{1}{4}}$ by r bits, one has to do an additional computation of the complexity of an exhaustive search over $2r$ bits.

In the language of lattices this becomes immediately clear. With Γ as above and $\mathbf{v} = \mathbf{0}$ (as we're still in the homogeneous case), the results of Section 3.2 show that for $\delta = \beta = \frac{1}{4} + \epsilon$, the complexity of the attack is $O(N^{2\epsilon}(\log N)^2)$.

The example given in [11] will go as follows in our method. We start with the lattice

$$\Gamma = \begin{pmatrix} 2^{38} & 0 \\ e & N \end{pmatrix} = \begin{pmatrix} & 2^{38} & & 0 \\ 711516780480876521042731877667548624237348233 & & & \end{pmatrix}$$

(note that in [11] the value of e contains a misprint).

We compute the reduced basis

$$\Gamma_{\text{red}} = \begin{pmatrix} 42694311384449024 & 87227281088446464 \\ 34997160860155755 & -133735834148055649 \end{pmatrix}.$$

The lattice point we need is $\Gamma \begin{pmatrix} 2d \\ -k \end{pmatrix} = \Gamma \begin{pmatrix} 3295186 \\ -735493 \end{pmatrix} = \Gamma_{\text{red}} \begin{pmatrix} 11 \\ 5 \end{pmatrix}$. Here $2d$ appears instead of d because in [11] $ed \equiv 1 \pmod{\text{lcm}(p-1, q-1)}$ is taken, and in this case $\text{gcd}(p-1, q-1)$ appears to be equal to 2.

This shows that, at least in this example, the efficiency of our method is comparable to [11], since we had to search for the numbers 11 and 5 of resp. 3.5 and 2.3 bits, together less than 7 bits (rather than 6 bits, because we have to allow negative values for one of the coordinates).

The fact that Verheul and Van Tilborg require a computation of the complexity of a $2r$ bit exhaustive search to allow r unknown bits more than $\frac{1}{4}$ th of N for both d and the unknown part of d (which, in this case, are of course the same), corresponds to our complexity results of Section 3.2. However, it does not directly imply that their method can be used in a partial key exposure setting. In that sense our result, with the homogeneous case being a special case of the general case, implies the result of [11], but not the other way around. We believe that the method of Verheul and Van Tilborg can be combined with the method of Baker and Davenport [1], for solving inhomogeneous diophantine approximation problems, but we see no advantages above our uniform and clean lattice method.

Finally, we will show that the cases of Wiener and Verheul/Van Tilborg are *deterministic* situations in our method.

Recall that we look for a small pair (d, k) such that

$$\begin{pmatrix} C & 0 \\ e & N \end{pmatrix} \begin{pmatrix} d \\ -k \end{pmatrix} = \begin{pmatrix} Cd \\ ed - kN \end{pmatrix} \approx \begin{pmatrix} N^{\beta+\frac{1}{2}} \\ N^{\beta+\frac{1}{2}} \end{pmatrix}.$$

We will argue that if $d < N^{\frac{1}{4}}$ (Wiener’s case), this small vector is actually the smallest nonzero lattice vector, which will be found by the Lagrange reduction.

Suppose it is not the smallest vector. Then the smallest vector cannot be linearly independent from it, for else the product of their sizes is smaller than $N^{2\beta+1} < N^{\frac{3}{2}}$, whereas the determinant of the lattice is $\det(L) = CN = N^{\frac{3}{2}}$.

This is a contradiction. The other option when $\begin{pmatrix} Cd \\ ed - kN \end{pmatrix}$ is not the smallest vector, is that the smallest vector is

$$\begin{pmatrix} Cx \\ ex - yN \end{pmatrix} = \alpha \begin{pmatrix} Cd \\ ed - kN \end{pmatrix}, \text{ for some } \alpha \in [-1, 1].$$

It follows that $d = \frac{1}{\alpha}x$ and $k = \frac{1}{\alpha}y$, and since $ed - k\phi(N) = 1$, it must hold that

$$ex - y\phi(N) = \alpha.$$

Since the left hand side is an integer, $\alpha \neq 0$, and $\alpha \in [-1, 1]$, it follows that $|\alpha| = 1$. Therefore, $d = |x|$ and $k = |y|$. Hence, the shortest reduced basis vector immediately gives us d and k . Thus, the method is clearly deterministic.

In the case of Verheul/Van Tilborg’s attack, $d = N^{\frac{1}{4}+\epsilon}$, so

$$\begin{pmatrix} Cd \\ ed - kN \end{pmatrix} \approx \begin{pmatrix} N^{\beta+\frac{1}{2}} \\ N^{\beta+\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} N^{\frac{3}{4}+\epsilon} \\ N^{\frac{3}{4}+\epsilon} \end{pmatrix},$$

so this vector is not the smallest reduced vector. However, one can see that the smallest vector must be linearly independent of it, so we know that

$$a^{-1} \det(L)^{\frac{1}{2}} \cdot N^{\frac{3}{4}+\epsilon} \geq \det(L).$$

It follows that $a < \det(L)^{-\frac{1}{2}} N^{\frac{3}{4}+\epsilon} = \det(L)^{-\frac{1}{2}} N^{\beta+\frac{1}{2}} = N^{\frac{1}{2}(\beta+\delta-\frac{1}{2})}$ and from the computations in Section 3.1, we know that this means that the search area is $O(N^{\beta+\delta-\frac{1}{2}}) = O(N^{2\epsilon})$. So one can see that in this case, one also does not depend on Assumption 1.

4.2 Comments on the Size of a in Other Cases

When we are outside the regions where the known continued fractions methods from Wiener and Verheul/Van Tilborg apply, the attack depends on Assumption 1, namely that the elements of Γ_{red} are all of size $\det(L)^{\frac{1}{2}}$. In this section, we will comment on how this assumption holds in practice.

Let m be the maximal entry of Γ_{red} , and $m = a \det(L)^{\frac{1}{2}}$. We want to check that for the matrices involved in the attacks of this paper, a is close to 1. Therefore, we performed tests for the attacks for small d in the following setup: N is an 2048 bit modulus, $\beta \in [0.25, 0.5]$, $\epsilon \in [0, 0.1]$, and $\delta = \text{Min}\{\beta, \frac{1}{2} - \beta + \epsilon\}$.

For this case, the lattices behaved as expected. In 500 experiments, the average value of a was approximately 1.9, and the maximal value of a was approximately 39.

5 Efficiency of the Attack

Now let us give some intuition on how our attack compares in running time to the other known results on partial key exposure attacks on small d , by Ernst et al. [7].

Figure 5 and 6 are two pictures of the attacks that are currently known and that use knowledge of MSBs or LSBs of d for relatively small d . The pictures show, for each value of β (the size parameter of d) what fraction of d we need to know in order to mount a successful attack. The area where the attack of this paper applies is dark.

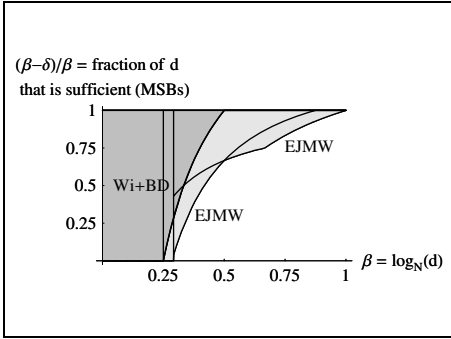


Fig. 5. Small d with known MSBs

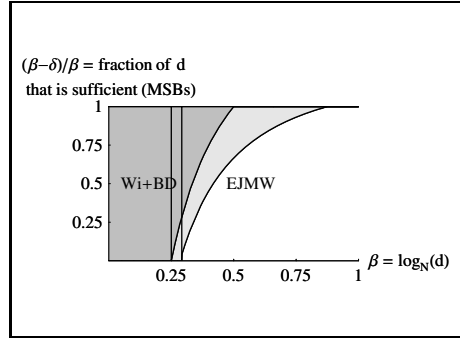


Fig. 6. Small d with known LSBs

One can see that our results do not exceed or match the optimal bounds which are already available. This is not surprising, since the attacks of [7] use large lattices, containing shifts of the RSA key equation and therefore combinations of monomials. Besides that, the attacks of [7] have asymptotic bounds. Hence, in those cases, to be able to perform close to the theoretic bound, there can be immensely large lattices involved, of which the reduction takes hours, days, or longer. Our attack belongs to those situations of partial key exposure that require only the reduction of a 2-dimensional lattice to solve, which an attacker can perform in just a few seconds. Moreover we can even exceed the theoretical bounds of the attacks with a small value ϵ .

To give some intuition of what this means in practice, we matched the 2-dimensional attack on small d and known MSBs against those of Ernst et al. [7]. The result is shown in the following table. For different values of β and δ , and different moduli N of 2048 bits, we computed the time to perform an attack for both methods.

This time includes:

- lattice reduction, resultant computations, and using the root $p + q - 1$ of the resultant polynomials to find p and q , for [7],
- lattice reduction and trying all pairs (z_1, z_2) to find p, q , for the 2-dimensional method.

In the table, it shows that for $\beta = 0.30$, and $\delta = 0.205$, our attack works in approximately 2 seconds (this is an average over 50 experiments). It uses a simple Mathematica program that runs on a computer with Pentium III processor of 733 MHz.

On the other hand, for the same parameters we need about 40 minutes to solve the problem using one of the methods of [7], and the smallest lattice for which their attack works is of dimension 30 in this case. These experiments were done using Shoup’s Number Theory Library [10], on a shared server with a Pentium IV Xeon processor of 2.80 GHz.

For the cases $\{\beta = 0.30, \delta = 0.210\}$, $\{\beta = 0.35, \delta = 0.150\}$, and $\{\beta = 0.35, \delta = 0.160\}$, one can see from the table that there are ‘breaking points’ for the methods

β	δ	Dim. lattice [7]	Time method [7]	Time 2D-method
0.30	0.050	10	35 sec.	1 sec.
0.30	0.100	10	35 sec.	1 sec.
0.30	0.150	10	35 sec.	1 sec.
0.30	0.200	30	40 min.	1 sec.
0.30	0.205	30	40 min.	2 sec.
0.30	0.210	30 / 50	40 min. / $4\frac{1}{2}$ hrs.	21 min.
0.35	0.050	10	35 sec.	1 sec.
0.35	0.100	10	35 sec.	1 sec.
0.35	0.150	14 / 30	1 min. / 40 min.	1 sec.
0.35	0.155	30	40 min.	2 sec.
0.35	0.160	30 / 50	40 min. / $4\frac{1}{2}$ hrs.	21 min.
0.40	0.050	10	35 sec.	1 sec.
0.40	0.100	14	1 min.	1 sec.
0.40	0.105	14	1 min.	2 sec.
0.40	0.110	14	1 min.	21 min.
0.45	0.050	14	1 min.	1 sec.
0.45	0.055	14	1 min.	2 sec.
0.45	0.060	14	1 min.	21 min.

Fig. 7. Experimental results: Comparison with [7]

of Ernst et al. For instance, if $\beta = 0.30$ and $\delta = 0.210$, the 30-dimensional lattice attack of [7] might suffice in some situations, whereas in others it will not lead to the solution. Therefore, for these parameters, it is possible that the attack takes either 40 minutes (if the attack using the 30-dimensional lattice works), or approximately $4\frac{1}{2}$ hours (if the 30-dimensional attack does not work and one has to use the 50-dimensional lattice attack).

6 Conclusion

We have shown how to perform a partial key exposure attack on RSA using a 2-dimensional lattice. The attack applies when the private exponent d is chosen to be small, which occurs in practice. In most cases, the attack is heuristic, but the underlying assumption is a reasonable one and supported by experiments. Although the attack does not achieve the theoretic bounds of known partial key exposure attacks using Coppersmith’s method, it is much faster in the area where it applies. Moreover, the attack shows what you can achieve with the simplest lattices possible, and also provides a link with the known attacks based on continued fractions techniques, as they appear as special deterministic cases of our attack.

References

1. A. BAKER, H. DAVENPORT, "The equations $3x^2 - 2 = y^2$ and $8x^2 - 7 = z^2$ ", *Quarterly Journal of Mathematics (Oxford) (2)* **20** [1969], pp. 129–137.
2. DAN BONEH, "Twenty years of Attacks on the RSA Cryptosystem", *Notices of the American Mathematical Society* **46** [1999], pp. 203–213.

3. DAN BONEH, GLENN DURFEE, "Cryptanalysis of RSA with Private Key d less than $N^{0.292}$ ", *IEEE Transactions on Information Theory* **46** [2000], pp. 1339–1349.
4. DAN BONEH, GLENN DURFEE, YAIR FRANKEL, "An Attack on RSA given a Small Fraction of the Private Key Bits", *Proceedings of ASIACRYPT 1998, LNCS 1514* [1998], pp. 25–34.
5. JOHANNES BLÖMER, ALEXANDER MAY, "New Partial Key Exposure Attacks on RSA", *Proceedings of CRYPTO 2003, LNCS 2729* [2003], pp. 27–43.
6. DON COPPERSMITH, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities", *Journal of Cryptology* **10** [1997], pp. 233–260.
7. MATTHIAS ERNST, ELLEN JOCHEMSZ, ALEXANDER MAY, BENNE DE WEGER, "Partial Key Exposure Attacks on RSA up to Full Size Exponents", *Proceedings of EUROCRYPT 2005, LNCS 3494* [2005], pp. 371–386.
8. ARJEN LENSTRA, HENDRIK LENSTRA, JR., LÁSZLÓ LOVÁSZ, "Factoring Polynomials with Rational Coefficients", *Mathematische Annalen* **261** [1982], pp. 515–534.
9. ALEXANDER MAY, "New RSA Vulnerabilities Using Lattice Reduction Methods", *PhD Thesis, University of Paderborn* [2003]
10. VICTOR SHOUP, "Online Number Theory Library", <http://www.shoup.net/ntl>
11. ERIK VERHEUL, HENK VAN TILBORG, "Cryptanalysis of 'less short' RSA Secret Exponents", *Applicable Algebra in Engineering, Communication and Computing* **8** [1997], pp. 425–435.
12. BENNE DE WEGER, "Algorithms for Diophantine Equations", *CWI Tract 65, Centre for Mathematics and Computer Science, Amsterdam* [1989].
13. MICHAEL WIENER, "Cryptanalysis of Short RSA Secret Exponents", *IEEE Transactions on Information Theory* **36** [1990], pp. 553–558.

On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search

Joonsang Baek^{1,*}, Reihaneh Safavi-Naini², and Willy Susilo²

¹ Institute for Infocomm Research, Singapore
jsbaek@i2r.a-star.edu.sg

² University of Wollongong, Australia
{rei, wsusilo}@uow.edu.au

Abstract. In this paper, we consider the problem of combining a public key encryption (PKE) scheme and a public key encryption with keyword search (PEKS) scheme proposed by Boneh, Di Crescenzo, Ostrovsky and Persiano (BDOP) in Eurocrypt 2004. We argue that the two schemes need to be treated as a single scheme to securely provide the PEKS service that BDOP envisioned. We formally define such a scheme, which we call “PKE/PEKS” and its security against chosen ciphertext attack, which we call “IND-PKE/PEKS-CCA”. We then construct a highly efficient PKE/PEKS scheme using the PEKS scheme presented by BDOP and a variation of ElGamal encryption scheme and show that it is IND-PKE/PEKS-CCA secure in the random oracle model assuming that the Computational Diffie-Hellman (CDH) problem is intractable. We also propose a generic construction of PKE/PEKS, which is slightly less efficient than the first one. Finally, we present two extensions of a PKE/PEKS scheme to the multi-receiver setting and multi-keyword setting.

1 Introduction

1.1 Motivation

Suppose that Bob’s email is encrypted under Alice’s public key and the encrypted email is stored in Alice’s email server. The *public key encryption with keyword search (PEKS)* scheme proposed by Boneh et al. [4] enables the server to test whether the encrypted email contains a particular keyword that Alice has requested and routes the email accordingly.

Following the descriptions given in [4], we describe this mechanism more precisely as follows. Let (pk, sk) be Alice’s public/private key pair. Bob encrypts his message (email) m using an encryption algorithm E of the public key encryption (PKE) scheme under Alice’s public key pk . Bob also encrypts a keyword w using

* A part of this work was done while the first author was with the School of Information Technology and Computer Science, University of Wollongong, Australia.

PEKS algorithm of the PEKS scheme under Alice’s public key pk . The resulting ciphertext

$$C = c||\tau = E(pk, m)||\text{PEKS}(pk, w) \quad (1)$$

is sent to Alice’s email server. Upon receiving a “trapdoor” t_w associated with the keyword w , the server checks whether τ encrypts w and if it does, the server sends c to Alice, otherwise, it sends off a message informing a search error. (Note that since Alice knows sk , she will be able to decrypt c).

The basic security requirement of the above scheme is that τ does not reveal any information about the keyword w to the server unless it receives the trapdoor t_w from Alice. We emphasize, however, that this requirement is concerned *only* with the security of a PEKS part of the ciphertext (that is, τ in C) and is not concerned with the security of the *integrated* scheme that combines PKE and PEKS and outputs $C = c||\tau$ in Equation (1).

The security of the integrated scheme has practical importance as in reality, one cannot always assume that the server behaves honestly. For example, there is a chance that the server is broken into by malicious intruder Eve. Obviously, Eve can *delete* some or all of the ciphertexts entries (C ’s), stored in the server so that Alice cannot receive the intended emails. This “deletion attack” is possible but the number of entries in the server can easily be recorded and audited later, and as a result the malicious behavior can be detected. What we focus on in this paper is a more sophisticated attack whereby Eve *modifies* $C (= c||\tau)$ ’s to deceive Alice or get useful information about the plaintext that c encrypts. An example of such attack is what we call, “swapping attack”. In this attack, Eve simply interchanges τ ’s so that Alice does not receive the correct message that Bob has sent to her. More precisely, Eve replaces τ in $c||\tau$ with τ' in $c'||\tau'$, where τ and τ' are PEKS for the keyword w and w' respectively, so that when Alice sends the trapdoor $t_{w'}$ (that corresponds to w') to the honest server, the server administrator will retrieve c (instead of c') for her. This attack is serious especially when Alice wants to retrieve the messages with high priority, for example, the messages associated with a keyword “urgent”. By launching this attack, Eve can *randomize* priority so that Alice gets the messages that are not really urgent. In addition to this swapping attack, a more active attack can be considered based on the fact that PEKS is a *public key* algorithm, so Eve can generate a PEKS ciphertext of a keyword of her choice. That is, Eve can modify $c||\tau$ by creating a new PEKS τ'' and replaces τ with it.

An immediate solution for preventing PEKS from the above-described attacks might be providing an authentication tag by applying a message authentication code (MAC) scheme, together with a shared key between the sender and receiver, to τ . However, we do not favor this trivial solution as this method will destroy the asymmetric nature of PEKS. One may then consider applying a digital signature scheme to τ but in fact, this method will not prevent the above-described attacks since anyone who has obtained τ can create a valid signature on it. Difficulties in providing authentication for PEKS will further be discussed in Section 4.1.

1.2 Related Work

The attacks illustrated in the previous section show that PEKS should not be treated as a stand-alone scheme, and it is important to carefully define and analyze a scheme in which PEKS is combined with PKE. However, there are few papers that discuss this issue in the context of PEKS. Boneh, Di Crescenzo, Ostrovsky and G. Persiano (BDOP) [4] briefly mention without giving an actual implementation that Equation (1) is not secure against *chosen ciphertext attack* but it can be made to be so using the techniques of [5]. However, as widely known, the techniques of [5] are based on non-interactive zero-knowledge proof, so the resulting scheme may not be efficient enough to be used in practice. Different from BDOP, our approach is more practice-oriented: We use a special property of ElGamal encryption to provide chosen ciphertext security for the scheme that combines PKE and PEKS, which actually prevents the attacks informally described in the previous section.

To our knowledge, other research papers dealing with PEKS including [11], [10] and [9] do not elaborate on the above issues. Most recently, Park, Cha and Lee (PCL) [8] has proposed a scheme that integrates PKE and PEKS, which they call “Searchable Keyword-Based Encryption (SKBE)”. They provide a chosen ciphertext security notion called “IND-SKBE-CCA” but this is different from our security notion which will be presented in the later section in that SKBE allows Alice to use a delegatable private key called “decrypt trapdoor” derived from her private key to decrypt the ciphertext.

1.3 Our Contributions

In this paper, we elaborate on the issues regarding integration of a PKE and PEKS. We formally define a combined scheme for PKE and PEKS, which we call “PKE/PEKS” and its security against chosen ciphertext attack, which we call “IND-PKE/PEKS-CCA”. We then provide a highly efficient construction of PKE/PEKS based on the PEKS scheme presented in [4] and the variation of ElGamal encryption scheme with the randomness reuse technique [6] and show that it is IND-PKE/PEKS-CCA secure in the random oracle model [3] assuming that the Computational Diffie-Hellman (CDH) problem is computationally hard. Additionally, we present another construction of PKE/PEKS scheme which is generic but slightly less efficient than the first one. We also present two extensions of the proposed PKE/PEKS scheme to multi-receiver setting and multi-keyword setting.

2 Preliminaries

2.1 Symbols and Notations

We use the notation $A(\cdot, \dots, \cdot)$ to denote an algorithm (modeled as a probabilistic Turing machine), with input arguments separated by commas. The notation $A^{O(\cdot)}(\cdot, \dots, \cdot)$ denotes that algorithm A makes calls to an oracle $O(\cdot)$. We

use $a \leftarrow A(x_1, \dots, x_n)$ to denote the assignment of a uniformly and independently distributed random element from the output of A on input (x_1, \dots, x_n) to the variable a . Given a set S , we denote by $b \stackrel{R}{\leftarrow} S$ the assignment of a uniformly and independently distributed random element from the set S to the variable b .

2.2 Formal Definition of PEKS and Description of BDOP-PEKS

We review the formal definition of PEKS given in [4]. In PEKS, three parties, which we call “sender”, “receiver” and “server”, are involved. (One can think the server as a physical storage device and its administrator). Once the sender sends a “PEKS ciphertext” which encrypts a keyword using the receiver’s public key, it is stored in the server. Upon receiving a trapdoor associated with a particular keyword, the server can check whether its one or some PEKS ciphertexts indeed encrypt the keyword that the receiver is referring to. A formal definition is as follows.

Definition 1 (PEKS). A public key encryption with keyword search (PEKS) scheme consists of the following algorithms.

- **KeyGen**(k): Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm generates a private and public key pair (sk, pk) of the receiver. Note that pk includes the security parameter k and the description of a finite keyword space SP_w . We write $(sk, pk) \stackrel{R}{\leftarrow} \text{KeyGen}(k)$.
- **PEKS**(pk, w): Taking the receiver’s public key pk and a keyword w as input, this algorithm generates a PEKS ciphertext τ which encrypts w . We write $\tau \leftarrow \text{PEKS}(pk, w)$.
- **Trapdoor**(sk, w): Taking a private key sk and a keyword w as input, this algorithm generates a trapdoor t_w for the keyword w . We write $t_w \leftarrow \text{Trapdoor}(sk, w)$.
- **Test**(t_w, τ): Taking a trapdoor t_w for a keyword w and a PEKS ciphertext $\tau = \text{PEKS}(pk, w')$, this algorithm returns a symbol “yes” if $w = w'$ and “no” otherwise.

The security notion for PEKS, called “semantic security of PEKS against chosen keyword attack” [4] concerns the confidentiality of keyword: Informally, if a PEKS scheme is secure against chosen keyword attack, the attacker who is able to query a number of keywords to the **Trapdoor** oracle should not be able to distinguish an encryption of a keyword w_0 from an encryption of a keyword w_1 for which he did not obtain the trapdoor.

Figure 1 describes the first PEKS scheme based on the bilinear pairings proposed by Boneh et al. [4], which we call “BDOP-PEKS”. Note that the bilinear pairing e that will be used throughout this paper is the *admissible* bilinear pairing, which is defined over two groups of the same prime-order q denoted by \mathbb{G}_1 and \mathbb{G}_2 . Suppose that \mathbb{G}_1 is generated by g . Then, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ has the following properties: 1) Bilinear: $e(g^a, g^b) = e(g, g)^{ab}$, for all $a, b \in \mathbb{Z}_q$ and 2) Non-degenerate: $e(g, g) \neq 1$.

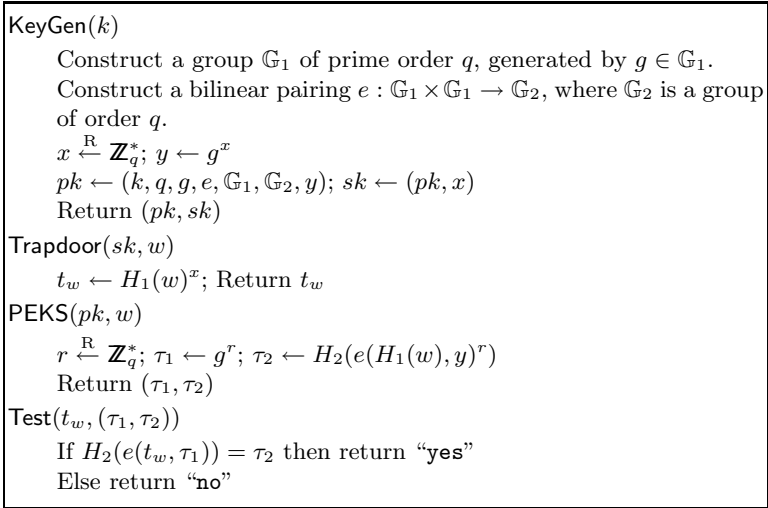


Fig. 1. BDOP-PEKS

3 PKE/PEKS

3.1 Formal Model

Recall that simply appending a PEKS ciphertext to a ciphertext that encrypts a message is vulnerable to the attacks described in Section 1.1. These attacks suggest that some mechanism should be provided to *bind* the ciphertext c that encrypts a message m and the PEKS ciphertext τ that encrypts a keyword w in such a way that any alteration of c and τ en route to B should be detected. We call this mechanism “tagging”. We argue that this tagging is essential in providing the PEKS service that BDOP originally envisioned [4]. Definition 2 presents formal description of a scheme that combines PKE and PEKS together with the tagging mechanism, which we call “PKE/PEKS”.

Definition 2 (PKE/PEKS). A PKE/PEKS scheme consists of the following algorithms.

- **KeyGen**(k): Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm generates a private and public key pair (sk, pk) . Note that pk includes the security parameter k and the description of a finite keyword space SP_w . We write $(sk, pk) \leftarrow \text{KeyGen}(k)$.
- **ENC-PKE/PEKS**(pk, w, m): Taking (pk, w, m) , where w and m denote a keyword and a plaintext respectively, as input, this algorithm generates a ciphertext c that encrypts m , a PEKS ciphertext τ that encrypts w and a tag σ . We write $(c, \tau, \sigma) \leftarrow \text{ENC-PKE/PEKS}(pk, w, m)$.
- **Trapdoor**(sk, w): Taking (sk, w) as input, this algorithm generates a trapdoor t_w for the keyword w . We write $t_w \leftarrow \text{Trapdoor}(sk, w)$.

- $\text{Test}(t_w, c, \tau, \sigma)$: Taking (t_w, c, τ, σ) as input, this algorithm returns (c, τ, σ) if τ encrypts w and “no” otherwise.
- $\text{DEC-PKE/PEKS}(sk, c, \tau, \sigma)$: Taking (sk, c, τ, σ) as input, this algorithm checks whether σ is a valid tag of c and τ . If it is, this algorithm outputs a decryption of c , which is a plaintext m or “reject”. Otherwise, this algorithm simply outputs “reject”. We write $m/\text{reject} \leftarrow \text{DEC-PKE/PEKS}(sk, c, \tau, \sigma)$.

We note that Test is run by the server, and when it finds that τ encrypts w , the server sends (c, τ, σ) to the receiver (instead of sending “yes” message). By running DEC-PKE/PEKS , the receiver can obtain the message that the sender originally sent.

3.2 Security Notion for PKE/PEKS

Notice that in the scheme of PKE/PEKS, PEKS is no longer regarded as a stand-alone scheme but a scheme that is run together with the data encryption and tagging mechanism. In terms of security, we of course require that a PEKS ciphertext part of the PKE/PEKS ciphertext (that is, τ in (c, τ, σ)) does not reveal any information about a keyword that it encrypts. This is covered by the “semantic security of PEKS against chosen keyword attack” [4] explained in Section 2.2. In addition to this notion, we define a new security notion for PKE/PEKS.

As mentioned earlier, the ciphertexts that encrypt a message and a keyword, that is (c, τ) should not be modified en route to the intended receiver. On top of this, confidentiality of the message m should be provided. In fact, these requirements are covered by the following security notion, which we call “IND-PKE/PEKS-CCA”.

Definition 3 (IND-PKE/PEKS-CCA). Let A be an attacker. Let $\text{PKE/PEKS} = (\text{KeyGen}, \text{ENC-PKE/PEKS}, \text{Trapdoor}, \text{Test}, \text{DEC-PKE/PEKS})$ be a PKE-PEKS scheme. Consider the following game.

Game $G^{\text{ind-pke/peks-cca}}(k)$

Phase 1: $(sk, pk) \leftarrow \text{KeyGen}(k)$

Phase 2: $(m_0, m_1, w) \leftarrow A^{\text{DEC-PKE/PEKS}(sk, \cdot, \cdot)}(pk)$ (Here, $|m_0| = |m_1|$, $w \neq m_0$ and $w \neq m_1$).

Phase 3: $\beta \stackrel{R}{\leftarrow} \{0, 1\}$

Phase 4: $(c, \tau, \sigma) \leftarrow \text{ENC-PKE/PEKS}(pk, w, m_\beta)$

Phase 5: $\beta' \leftarrow A^{\text{DEC-PKE/PEKS}(sk, \cdot, \cdot)}(pk, (c, \tau, \sigma))$

Note that in the above game, A is not allowed to query the target PKE/PEKS ciphertext (c, τ, σ) to the oracle $\text{DEC-PKE/PEKS}(sk, \cdot, \cdot)$.

We define A 's advantage as

$$\mathbf{Adv}^{\text{ind-pke/peks-cca}}(k) = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|.$$

A breaks IND-PKE/PEKS-CCA of PKE/PEKS with (t, q_D, ϵ) if and only if the advantage of A that makes q_D decryption queries is greater than ϵ within

running time t . The scheme PKE/PEKS is said to be (t, q_D, ϵ) -IND-PKE/PEKS-CCA secure if there is no attacker A that breaks IND-PKE/PEKS-CCA of PKE/PEKS with (t, q_D, ϵ) .

4 PKE/PEKS Constructions

4.1 Difficulties in Realizing Secure PKE/PEKS

Before describing our PKE/PEKS scheme, we show that constructing a secure PKE/PEKS scheme is not very straightforward as it seems to be. We note that Appendix ?? which presents our chosen ciphertext attack on Park et al.’s recent proposal [8] also shows that caution must exercised when constructing a scheme that combines PKE and PEKS.

Suppose that we use any ElGamal-like CCA-secure public key encryption scheme for encrypting a plaintext message m , the BDOP-PEKS scheme [4] for encrypting a keyword w and some Diffie-Hellman key exchange-style message authentication code for creating a tag σ . Assuming that the same public key $y = g^x$, where x is a private key, is used, one may create a PKE/PEKS scheme as follows:

$$(c, \tau, \sigma) = (c, (g^r, H_2(e(H_1(w), y)^r)), (g^{r'}, H_3(y^{r'}, c, \tau))), \tag{2}$$

where $r, r' \in \mathbb{Z}_q^*$ are chosen at random; c is a ciphertext that encrypts m , output by the CCA-secure public key encryption scheme; and H_1, H_2 and H_3 are appropriate hash functions.

One can notice that the security of Equation (2) can easily be broken by the following chosen ciphertext attack *even though the underlying encryption for the message m is CCA-secure*: Leave c as it is and modify τ to τ' (which is different from τ). Then create another tag σ' by choosing a new random element r'' from \mathbb{Z}_q^* and computing $(g^{r''}, H_3(y^{r''}, c, \tau'))$. Query $(c, \tau', \sigma') (\neq (c, \tau, \sigma))$ to the decryption oracle DEC-PKE/PEKS. Note that (c, τ', σ') will pass the tagging check in the decryption process as σ' is a valid tag (of c and τ') and hence will return m , which implies breaking the confidentiality!

Considering the above attack, one may attempt to construct an extended PEKS scheme using the randomness re-use technique [6] as follows:

$$(c, \tau, \sigma) = (c, (g^r, H_2(e(H_1(w), y)^r)), H_3(y^r, c, \tau)), \tag{3}$$

where $r \in \mathbb{Z}_q^*$ is chosen at random.

However, Equation (3) also suffers from a similar problem as in the previous scheme as the attacker can create $\tau' = (g^{r'}, H_2(e(H_1(w), y)^{r'}))$ and $\sigma' = H_3(y^{r'}, c, \tau)$ for $r' \neq r$.

Intuitively, the above two attacks suggest that some secret key should be shared between c, τ and σ in such a way that any modification of them should result in invalid ciphertext which can be detected by the DEC-PKE/PEKS oracle. In the following sections, we show that this intuition actually works.

4.2 Construction Based on ElGamal/BDOP-PEKS

We use a PKE scheme similar to DHIES [1] or ElGamal version of REACT [7] and the BDOP-PEKS scheme (reviewed in Section 2.2) for our PKE/PEKS proposal. Importantly, we make use of the randomness re-use technique [6] (which is exploited more in [2]) to provide secure binding of PKE and PEKS ciphertexts and a high level of efficiency. In Figure 2, the proposed PKE/PEKS scheme is described.

<p>KeyGen(k)</p> <p>Construct a group \mathbb{G}_1 of prime order q, generated by $g \in \mathbb{G}_1$.</p> <p>Construct a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where \mathbb{G}_2 is a group of order q.</p> <p>Choose hash functions $H_1 : \mathbb{G}_1 \rightarrow \{0, 1\}^{l_1}$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$; $H_3 : \mathbb{G}_2 \rightarrow \{0, 1\}^{l_3}$; $H_4 : \{0, 1\}^* \rightarrow \{0, 1\}^{l_4}$</p> <p>$x \xleftarrow{R} \mathbb{Z}_q^*$; $y \leftarrow g^x$</p> <p>$pk \leftarrow (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, y)$; $sk \leftarrow (pk, x)$</p> <p>Return (pk, sk)</p> <p>ENC-PKE/PEKS(pk, w, m)</p> <p>$r \xleftarrow{R} \mathbb{Z}_q^*$; $c_1 \leftarrow g^r$; $\kappa \leftarrow y^r$</p> <p>$K \leftarrow H_1(\kappa)$; $c_2 \leftarrow K \oplus m$</p> <p>$h \leftarrow H_2(w)$; $\mu \leftarrow e(h, y)^r$; $\tau \leftarrow H_3(\mu)$</p> <p>$\sigma \leftarrow H_4(\kappa, m, c_1, c_2, \tau)$</p> <p>Return (c_1, c_2, τ, σ)</p> <p>Trapdoor(sk, w)</p> <p>$t_w \leftarrow H_2(w)^x$; Return t_w</p> <p>Test($t_w, c_1, c_2, \tau, \sigma$)</p> <p>If $H_3(e(t_w, c_1)) = \tau$ then return (c_1, c_2, τ, σ)</p> <p>Else return “no”</p> <p>DEC-PKE/PEKS($sk, c_1, c_2, \tau, \sigma$)</p> <p>$\kappa \leftarrow c_1^x$; $K \leftarrow H_1(\kappa)$; $m \leftarrow K \oplus c_2$</p> <p>If $H_4(\kappa, m, c_1, c_2, \tau) = \sigma$ then return m</p> <p>Else return “reject”</p>

Fig. 2. Our PKE/PEKS Scheme Based on BDOP-PEKS/ElGamal

Now, we analyze the security of the above scheme. First, we review the definition of the Computational Diffie-Hellman (CDH) problem.

Definition 4 (CDH). Let p and q be primes such that $q|p - 1$. Let g be a generator of \mathbb{G}_1 . (We assume that a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is defined). Let A be an attacker. A tries to solve the following problem: *Given (g, g^a, g^b) for uniformly chosen $a, b \in \mathbb{Z}_q^*$, compute $\kappa = g^{ab}$.*

Formally, we define A 's advantage by $\Pr[A(g, g^a, g^b) = g^{ab}]$. A solves the CDH problem with (t', ϵ') if and only if the advantage of A is greater than ϵ' within running time t' . The CDH problem is said to be (t', ϵ') -intractable if there is no attacker A that solves the CDH problem with (t', ϵ') .

In the following theorem, we prove that our scheme described in Figure 2 is IND-PKE/PEKS-CCA secure.

Theorem 1. *The PKE-PEKS scheme based on ElGamal, BDOP-PEKS and the randomness re-use technique, presented in Figure 2, is $(t, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_D, \epsilon)$ -IND-PKE/PEKS-CCA secure in the random oracle model assuming that the CDH problem is (t', ϵ') -intractable, where $\epsilon' > \epsilon - \frac{q_D}{2^{l_4}}$ and $t' = t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4})O(1) + (q_{H_1} + q_{H_4})(T_e + O(1)) + q_D(T_e + O(1))$ where q_{H_1}, \dots, q_{H_4} denote the number of queries to the random oracles H_1, \dots, H_4 respectively and T_e denotes the execution time for computing bilinear pairing.*

Proof. Let A be an IND-PKE/PEKS-CCA attacker. (The number of queries to the oracles that A makes and its running time are as defined in the above theorem statement). We show that using A , one can construct an attacker B that can solve the CDH problem.

Suppose that B is given $(p, q, g, \mathbb{G}_1, \mathbb{G}_2, g^a, g^b)$ as an instance of the CDH problem. (Note that B can test whether a given tuple is Diffie-Hellman one or not using the bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$). B can simulate the Challenger's execution of each phase of IND-PKE/PEKS-CCA game for A as follows.

[Simulation of Phase 1] B sets $y = g^b$ and gives A $(p, q, g, \mathbb{G}_1, \mathbb{G}_2, y)$ and simulate A 's queries to the random oracles H_1, H_2, H_3 , and H_4 as follows.

On receiving a query κ to H_1 :

1. If $e(g, \kappa) = e(g^a, y)$ then stop the simulation. (This means that κ is a Diffie-Hellman key of g^a and $y(= g^b)$).
2. Else do the following:
 - (a) If $\langle \kappa, K \rangle$ exists in $H_1\text{List}$ then return K as answer.
 - (b) Else pick $K \in \{0, 1\}^{l_1}$ at random, add $\langle \kappa, K \rangle$ to $H_1\text{List}$ and return K as answer.

On receiving a query w to H_2 :

1. If $\langle w, h, z \rangle$ exists in $H_2\text{List}$ then return h as answer.
2. Else pick $z \in \mathbb{Z}_q^*$ at random, add $\langle w, h, z \rangle$ to $H_2\text{List}$ and compute $h = g^z$ and return h as answer.

On receiving a query μ to H_3 :

1. If $\langle \mu, \tau \rangle$ exists in $H_3\text{List}$ then return τ as answer.
2. Else pick $\tau \in \{0, 1\}^{l_3}$ at random, add $\langle \mu, \tau \rangle$ to $H_3\text{List}$ and return τ as answer.

On receiving a query $(\kappa, m, c_1, c_2, \tau)$ to H_4 :

1. If $e(g, \kappa) = e(g^a, y)$ then stop the simulation. (This means that κ is a Diffie-Hellman key of g^a and $y(=g^b)$).
2. Else do the following:
 - (a) If $\langle(\kappa, m, c_1, c_2, \tau), \sigma\rangle$ exists in $\mathbf{H}_4\text{List}$ then return c_3 as answer.
 - (b) Else pick $\sigma \in \{0, 1\}^{l_4}$ at random, add $\langle(\kappa, m, c_1, c_2, \tau), \sigma\rangle$ to $\mathbf{H}_4\text{List}$ and return σ as answer.

[Simulation of Phase 2] B answers A 's decryption queries as follows. (B answers A 's other queries as described in Phase 1).

On receiving a decryption query (c_1, c_2, τ, σ) :

1. Search $\mathbf{H}_4\text{List}$ for a tuple $\langle(\kappa, m, c_1, c_2, \tau), \sigma\rangle$.
2. If such a tuple exists then do the following:
 - (a) If $e(g, \kappa) = e(c_1, y)$ then run the above H_1 oracle simulator to get a tuple $\langle\kappa, K\rangle \in \mathbf{H}_1\text{List}$ and checks whether $c_2 = m \oplus K$. If the equality holds then return m and “**Reject**” otherwise.
3. Else return “**Reject**”.

[Simulation of Phase 3] B creates a target ciphertext as follows.

On receiving a challenge query $((m_0, m_1), w^*)$, where $|m_0| = |m_1|$ and $w^* \neq m_0$ and $w^* \neq m_1$:

1. Run the above H_2 oracle simulator taking w^* as input to get a tuple $\langle w^*, h^*, z^* \rangle \in \mathbf{H}_2\text{List}$.
2. Compute $\mu^* = e(g^b, g^a)^{z^*}$ and run the H_3 simulator to get a tuple $\langle \mu^*, \tau^* \rangle \in \mathbf{H}_3\text{List}$.
3. Pick $c_2^* \in \{0, 1\}^{l_1}$, $\sigma^* \in \{0, 1\}^{l_4}$ and $\beta \in \{0, 1\}$ at random.
4. Set $c_1^* = g^a$ and $K^* = c_2^* \oplus m_\beta$.
5. Define $K^* = H_1(\kappa^*)$ and $\sigma^* = H_2(\kappa^*, m_\beta, c_1^*, c_2^*, \tau^*)$. (Note that B does not know the value κ^* yet).
6. Update $\mathbf{H}_4\text{List}$ by adding $\langle(-, m_\beta, c_1^*, c_2^*, \tau^*), \sigma^*\rangle$ as entry.
7. Return $(c_1^*, c_2^*, \tau^*, \sigma^*)$ as a target ciphertext.

[Simulation of Phase 4] In this phase, B answers A 's queries to the random oracle H_1 and H_4 using the target ciphertext $(c_1^*, c_2^*, \tau^*, \sigma^*)$ as follows. Note that the simulation for the random oracles H_2 and H_3 remains the same as Phase 1.

On receiving a query κ to H_1 :

1. If $e(g, \kappa) = e(c_1^*, y)$ then return K^* (set in the simulation of Phase 3) as answer and stop the simulation (since κ is a Diffie-Hellman key of $c_1^*(=g^a)$ and $y(=g^b)$).
2. Else do the following:
 - (a) If $\langle\kappa, K\rangle$ exists in $\mathbf{H}_1\text{List}$ then return K as answer.
 - (b) Else pick $K \in \{0, 1\}^{l_1}$ at random, add $\langle\kappa, K\rangle$ to $\mathbf{H}_1\text{List}$ and return K as answer.

On receiving a query $(\kappa, m, c_1, c_2, \tau)$ to H_4 :

1. If $(m, c_1, c_2, \tau) = (m_\beta, c_1^*, c_2^*, \tau^*)$ then do the following:
 - (a) If $e(g, \kappa) = e(c_1^*, y)$ then return σ^* (set in the simulation of Phase 3) as answer and stop the simulation (since κ is a Diffie-Hellman key of $c_1^*(=g^a)$ and $y(=g^b)$).
2. Else do the following:
 - (a) If $\langle(\kappa, m, c_1, c_2, \tau), \sigma\rangle$ exists in $H_4\text{List}$ then return c_3 as answer.
 - (b) Else pick $\sigma \in \{0, 1\}^{l_4}$ at random, add $\langle(\kappa, m, c_1, c_2, \tau), \sigma\rangle$ to $H_4\text{List}$ and return σ as answer.

[Simulation of Phase 5] When A outputs its β' , B stops the whole game.

[Analysis] We first evaluate the simulations of the random oracles given above. From the construction of H_1 and H_4 , it is clear that the simulations of H_1 and H_4 are perfect as long as A does not query κ^* to H_1 nor query $(\kappa^*, m, c_1, c_2, \tau)$ to H_4 , where κ^* is a Diffie-Hellman key of g^a and g^b . By AskH_1^* and AskH_4^* we denote the events that such κ^* has been queried to H_1 and H_4 respectively. Notice that the simulations of the random oracles H_2 and H_3 are perfect.

Next, one can notice that the simulated target ciphertext is identically distributed as the real one from the construction.

Now, we evaluate the simulation of the decryption oracle. We note that simulation errors may occur while B is running the decryption oracle simulator specified above. Suppose that (c_1, c_2, τ, σ) has been submitted as a *valid* decryption query to the decryption oracle simulator. Even if (c_1, c_2, τ, σ) is valid, there is a possibility that it can be rejected by the decryption oracle simulator. According to its construction, the decryption oracle simulator will reject the ciphertext unless $(\kappa, m, c_1, c_2, \tau)$ has been queried to H_4 . There are two cases when this event occurs:

- Case 1: σ^* (output of H_4) has been obtained from the target ciphertext.
- Case 2: The value for $H_4(\kappa, m, c_1, c_2, \tau)$ has been correctly guessed without invoking H_4 .

Case 1 leads to a contradiction as $(\kappa, m, c_1, c_2, \tau)$ must be the same as $(\kappa^*, m_\beta, c_1^*, c_2^*, \tau^*)$ by the collision-free property of H_4 . (Note that the target ciphertext is not allowed to be queried to the decryption oracle). Case 2 occurs only with negligible probability of $1/2^{l_4}$.

Let DecErr be an event that a valid ciphertext is rejected during the entire attack game. Then, since q_D decryption oracle queries are made, we have $\Pr[\text{DecErr}] \leq \frac{q_D}{2^{l_4}}$.

Now define an event E to be $\text{AskH}_1^* \vee \text{DecErr}$. If E does not happen, it is clear that A does not gain any advantage greater than $1/2$ to guess β due to the randomness of the output of the random oracle H_1 . Namely, we have $\Pr[\beta' = \beta | \neg E] \leq \frac{1}{2}$. Hence, by splitting $\Pr[\beta' = \beta]$, we obtain $\Pr[\beta' = \beta] = \Pr[\beta' = \beta | \neg E] \Pr[\neg E] + \Pr[\beta' = \beta | E] \Pr[E] \leq \frac{1}{2} \Pr[\neg E] + \Pr[E] = \frac{1}{2} + \frac{1}{2} \Pr[E]$ and $\Pr[\beta' = \beta] \geq \Pr[\beta' = \beta | \neg E] \Pr[\neg E] = \frac{1}{2} - \frac{1}{2} \Pr[E]$.

By definition of ε , we then have $\varepsilon < |\Pr[\beta' = \beta] - \frac{1}{2}| \leq \frac{1}{2} \Pr[E] \leq \Pr[\text{AskH}_1^* \vee \text{DecErr}] \leq \Pr[\text{AskH}_1^*] + \Pr[\text{DecErr}]$. Since $\Pr[\text{DecErr}] \leq \frac{q_D}{2^{l_4}}$, we obtain $\Pr[\text{AskH}_1^*] > \varepsilon - \frac{q_D}{2^{l_4}}$. Meanwhile, if AskH_1^* happens then B will be able to solve the CDH problem, that is, $\Pr[\text{AskH}_1^*] = \varepsilon'$. Thus we obtain $\varepsilon' > \varepsilon - \frac{q_D}{2^{l_4}}$.

The running time of the CDH attacker B is $t' > t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4})O(1) + (q_{H_1} + q_{H_4})(T_e + O(1)) + q_D(T_e + O(1))$ where T_e denotes the execution time for pairing computation (to check whether a given tuple is a Diffie-Hellman one or not).

```

KeyGen(k)
    (pk', sk') ← KeyGen(k)
    Choose hash functions  $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{l_1}$  and
     $H_4 : \{0, 1\}^* \rightarrow \{0, 1\}^{l_4}$ 
    pk ← (pk',  $H_1, H_4$ ); sk ← sk'
    Return (pk, sk)
ENC-PKE/PEKS(pk, w, m)
    c1 ← E(pk, R); K ←  $H_1(R)$ 
    c2 ←  $K \oplus m$ 
     $\tau$  ← PEKS'(pk, w)
     $\sigma$  ←  $H_4(R, m, c_1, c_2, \tau)$ 
    Return (c1, c2,  $\tau$ ,  $\sigma$ )
Trapdoor(sk, w)
    tw ← Trapdoor'(sk, w)
Test(tw, (c1, c2,  $\tau$ ,  $\sigma$ ))
    If Test'(tw,  $\tau$ ) = "yes" return (c1, c2,  $\tau$ ,  $\sigma$ )
    Else return "no"
DEC-PKE/PEKS(sk, (c1, c2,  $\tau$ ,  $\sigma$ ))
    R ← D(sk, c1); K ←  $H_1(R)$ 
    m ←  $c_2 \oplus K$ ;
    If  $H_4(R, m, c_1, c_2, \tau) = \sigma$  then return m
    Else return "reject"
    
```

Fig. 3. Generic Construction of PKE/PEKS

4.3 Generic Construction

Our second construction of the PKE/PEKS scheme is based on more generic primitives. Let $\text{PKE} = (\text{KeyGen}, \text{E}, \text{D})$ be any public key encryption scheme secure in the OW-PCA (One Wayness under Plaintext Checking Attack) sense. (OW-PCA will be formally defined shortly). Let $\text{PEKS} = (\text{KeyGen}', \text{PEKS}', \text{Trapdoor}', \text{Test}')$ be any PEKS scheme in which public/private key pairs generated by KeyGen' are the same as the public/private key pairs generated by

KeyGen of PKE. Below in Figure 3, we describe our generic construction of PKE/PEKS.

Before analyzing the security of the scheme in Figure 3, we review the OW-PCA notion.

Definition 5 (OW-PCA). Let A be an attacker. Let $\text{PKE} = (\text{KeyGen}, \text{E}, \text{D})$ be a PKE scheme. Consider the following game.

Game $\text{G}^{\text{ow-pca}}(k)$

Phase 1: $(pk, sk) \leftarrow \text{KeyGen}(k)$

Phase 2: $m \leftarrow SP_m$ (SP_m denotes “message space”); $c \leftarrow \text{E}(pk, m)$

Phase 3: $m' \leftarrow A^{\text{PCO}(\cdot, \cdot)}(pk, c)$

Note that in the above game, PCO (Plaintext Checking Oracle) checks, given a message and ciphertext pair (m, c) , whether c encrypts m or not.

We define A 's advantage as

$$\text{Adv}^{\text{ow-pca}}(k) = \Pr[m' = m].$$

A breaks OW-PCA of PKE with (t, q_O, ϵ) if and only if the advantage of A that makes q_O PCO queries is greater than ϵ within running time t . The scheme PKE is said to be (t, q_O, ϵ) -OW-PCA secure if there is no attacker A that breaks OW-PCA of PKE with (t, q_O, ϵ) .

Regarding the second construction, we obtain the following theorem.

Theorem 2. *The generic construction of PKE-PEKS scheme presented in Figure 3 is $(t, q_{H_1}, q_{H_4}, q_D, \epsilon)$ -IND-PKE/PEKS-CCA secure in the random oracle model assuming that the underlying PKE encryption scheme PKE is (t', q_O, ϵ') -OW-PCA secure, where $\epsilon' > \epsilon - \frac{q_D}{2^4}$, $q_O = q_{H_1} + q_{H_4} + O(1)$ and $t' = t + (q_{H_1} + q_{H_4})O(1) + (q_{H_1} + q_{H_4})(T_O + O(1)) + q_D(T_O + O(1))$ where q_{H_1} and q_{H_4} denote the number of queries to the random oracles H_1 and H_4 respectively and T_O denotes the execution time for the PCO to check whether a given ciphertext encrypts a given plaintext.*

Due to lack of space, the proof is given in the full version of this paper.

5 Extensions to Multi-receiver Multiple Keywords Setting

Assume that the sender wants to send confidential messages to a group of receivers, to which a PEKS ciphertext that encrypts a keyword is attached. In this setting, we assume that the receivers have their own public keys denoted by pk_1, \dots, pk_n . Based on the PKE/PEKS scheme described in Figure 2, one can build a Multi-Receiver (MR)-PKE/PEKS scheme. See Figure 4.

```

KeyGen( $k$ )
  Construct a group  $\mathbb{G}_1$  of prime order  $q$ , generated by  $g \in \mathbb{G}_1$ .
  Construct a bilinear pairing  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ , where  $\mathbb{G}_2$  is a group
  of order  $q$ .
  Choose hash functions  $H_1 : \mathbb{G}_1 \rightarrow \{0, 1\}^{l_1}$ ,  $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ ;
   $H_3 : \mathbb{G}_2 \rightarrow \{0, 1\}^{l_3}$ ;  $H_4 : \{0, 1\}^* \rightarrow \{0, 1\}^{l_4}$ 
  For  $i = 1, \dots, n$  do
     $x_i \xleftarrow{R} \mathbb{Z}_q^*$ ;  $y_i \leftarrow g^{x_i}$ 
     $pk_i \leftarrow (k, q, g, e, \mathbb{G}_1, \mathbb{G}_2, y_i)$ ;  $sk_i \leftarrow (pk_i, x_i)$ 
  End For
   $\mathbf{pk} \leftarrow (pk_1, \dots, pk_n)$ ;  $\mathbf{sk} \leftarrow (sk_1, \dots, sk_n)$ 
  Return ( $\mathbf{pk}, \mathbf{sk}$ )

ENC-MR-PKE/PEKS( $\mathbf{pk}, w, \mathbf{m}(= (m_1, \dots, m_n))$ )
   $r \xleftarrow{R} \mathbb{Z}_q^*$ ;  $c_1 \leftarrow g^r$ 
  For  $i = 1, \dots, n$  do
     $\kappa_i \leftarrow y_i^r$ ;  $K_i \leftarrow H_1(\kappa_i)$ ;  $c_{2i} \leftarrow K_i \oplus m_i$ 
  End For
   $h \leftarrow H_2(w)$ ;  $\mu \leftarrow e(h, y)^r$ ;  $\tau \leftarrow H_3(\mu)$ 
  For  $i = 1, \dots, n$  do
     $\sigma_i \leftarrow H_4(\kappa_i, m_i, c_1, c_{2i}, \tau)$ 
  End For
  Return ( $c_1, c_{21}, \dots, c_{2n}, \tau, \sigma_1, \dots, \sigma_n$ )

Trapdoor( $sk, w$ )
   $t_w \leftarrow H_2(w)^x$ ; Return  $t_w$ 

Test( $t_w, c_1, c_{21}, \dots, c_{2n}, \tau, \sigma_1, \dots, \sigma_n$ )
  If  $H_3(e(t_w, c_1)) = \tau$  then return ( $c_1, c_{2i}, \tau, \sigma_i$ ) for some  $i \in [1, n]$ 
  Else return “no”

DEC-PKE/PEKS( $sk_i, c_1, c_{2i}, \tau, \sigma_i$ ) for some  $i \in [1, n]$ 
   $\kappa_i \leftarrow c_1^{x_i}$ ;  $K_i \leftarrow H_1(\kappa_i)$ ;  $m_i \leftarrow K_i \oplus c_{2i}$ 
  If  $H_4(\kappa_i, m_i, c_1, c_{2i}, \tau) = \sigma_i$  then return  $m_i$ 
  Else return “reject”

```

Fig. 4. Extension of Our PKE/PEKS Scheme to Multi-Receiver Setting

Notice that in Figure 4, the same random value r is used for encrypting all the messages (m_1, \dots, m_n) in the ENC-MR-PKE/PEKS algorithm, so the length of the resulting ciphertext has been reduced by factor n .

Another natural extension of the PKE/PEKS scheme in Figure 2 to multiple keyword setting can be considered. Suppose that multiple keywords denoted by w_1, \dots, w_n are associated with a message m and the receiver wants to find a ciphertext that encrypts m which matches *any* of w_1, \dots, w_n . On receiving trapdoor t_{w_i} for w_i , the server searches the ciphertext \mathbf{c} and sends it together with a PEKS component τ_i for w_i and a tag σ_i . In Figure 5, we describe this scheme. (The security analysis of this scheme is very similar to that of the PKE/PEKS scheme in Section 4.2 and hence is omitted).

<p>KeyGen(k) : Same as KeyGen of the PKE/PEKS scheme in Section 4.2</p> <p>ENC-PKE/PEKS(pk, w_1, \dots, w_n, m)</p> <p style="padding-left: 20px;">$r \xleftarrow{R} \mathbb{Z}_q^*$; $c_1 \leftarrow g^r$; $\kappa \leftarrow y^r$</p> <p style="padding-left: 20px;">$K \leftarrow H_1(\kappa)$; $c_2 \leftarrow K \oplus m$</p> <p style="padding-left: 20px;">For $i = 1, \dots, n$ do</p> <p style="padding-left: 40px;">$h_i \leftarrow H_2(w_i)$; $\mu_i \leftarrow e(h_i, y)^r$; $\tau_i \leftarrow H_3(\mu_i)$</p> <p style="padding-left: 40px;">$\sigma_i \leftarrow H_4(\kappa, m, c_1, c_2, \tau_i)$</p> <p style="padding-left: 20px;">End For</p> <p style="padding-left: 20px;">Return $(c_1, c_2, \tau_1, \sigma_1, \dots, \tau_n, \sigma_n)$</p> <p>Trapdoor($sk, w_i$) for $i = 1, \dots, n$</p> <p style="padding-left: 20px;">$t_{w_i} \leftarrow H_3(w_i)^x$; Return t_{w_i}</p> <p>Test($t_{w_i}, c_1, c_2, \tau_1, \sigma_1, \dots, \tau_n, \sigma_n$)</p> <p style="padding-left: 20px;">If there exists τ_i such that $H_3(e(t_{w_i}, c_1)) = \tau_i$ then return $(c_2, c_2, \tau_i, \sigma_i)$ for some $i \in [1, n]$</p> <p style="padding-left: 20px;">Else return “no”</p> <p>DEC-PKE/PEKS($sk, c_1, c_2, \tau_i, \sigma_i$) for some $i \in [1, n]$</p> <p style="padding-left: 20px;">$\kappa \leftarrow c_1^x$; $K \leftarrow H_1(\kappa)$; $m \leftarrow K \oplus c_2$</p> <p style="padding-left: 20px;">If $H_4(\kappa, m, c_1, c_2, \tau_i) = \sigma_i$ then return m</p> <p style="padding-left: 20px;">Else return “reject”</p>
--

Fig. 5. Extension of Our PKE/PEKS Scheme to Multi-Keyword Setting

Due to lack of space, the security notions for the above two extensions and their security analysis are given in the full version of this paper.

6 Concluding Remarks

In this paper, we discussed the security issues related to the integration of PEKS and PKE. We formalized a scheme that combines PKE and PEKS, called “PKE/PEKS” and formulated its security against chosen ciphertext attack. We proposed a provably-secure and highly efficient PKE/PEKS scheme based on ElGamal, BDOP-PEKS [4] and the randomness re-use technique [6]. We also proposed a generic construction PKE/PEKS. Finally, we considered two extensions of the proposed PKE/PEKS scheme to the multi-receiver and multi-keyword settings.

References

1. M. Abdalla, M. Bellare and P. Rogaway, *The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES*, In CT-RSA 2003, LNCS 2020, pp. 143–158, Springer-Verlag, 2001.
2. M. Bellare, A. Boldyreva and D. Pointcheval, *Multi-Recipient Encryption Schemes: Security Notions and Randomness Re-Use*, In PKC 2003, LNCS 2567, pp. 85–99, Springer-Verlag, 2003.

3. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In ACM-CCS, pp. 62–73, 1993.
4. D. Boneh, G. Di Crescenzo, R. Ostrovsky and G. Persiano, *Public Key Encryption with Keyword Search*, In Eurocrypt '04, LNCS 3027, pp. 506–522, Springer-Verlag, 2004.
5. D. Dolev, C. Dwork and M. Naor, *Non-Malleable Cryptography*, Siam J. on Computing 30(2), pp. 391-437, 2000.
6. K. Kurosawa, *Multi-recipient Public-Key Encryption with Shortened Ciphertext*, In PKC '02, LNCS 2274, pp. 48–63, Springer-Verlag, 2002.
7. T. Okamoto and D. Pointcheval, *REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform*, In CT-RSA '04, LNCS 2020, pp. 159–175, Springer-Verlag, 2004.
8. D. J. Park, J. Cha and P. J. Lee, *Searchable Keyword-Based Encryption*, In IACR ePrint Archive, Report 2005/367, 2005.
9. D. J. Park, K. Kim and P. J. Lee, *Public Key Encryption with Conjunctive Field Keyword Search*, In WISA '04, LNCS 3325, pp. 73–86, Springer-Verlag, 2004.
10. J. Malone-Lee, M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, G. Neven, P. Paillier and H. Shi. *Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions* In Crypto '05, LNCS, pp. 205–222. Springer-Verlag, 2005.
11. B. Waters, D. Balfanz, G. Durfee, and D. Smetters, *Building an Encrypted and Searchable Audit Log*, In Network and Distributed System Security Symposium (NDSS 2004), 2004.

Collusion-Free Policy-Based Encryption

Walid Bagga and Refik Molva

Institut Eurécom
Corporate Communications
2229, route des Crêtes B.P. 193
06904 Sophia Antipolis, France
{bagga, molva}@eurecom.fr

Abstract. A policy-based encryption scheme allows a user to encrypt a message with respect to a credential-based policy formalized as monotone boolean expression written in standard normal form. The encryption is so that only a user having access to a qualified set of credentials for the policy is able to successfully decrypt the message. An inherent property of policy-based encryption is that in addition to the recipient an encrypted message is intended for, any collusion of credential issuers or end users who are able to collect a qualified set of credentials for the policy used to encrypt the message can decrypt it as well. In some applications, the collusion property may be acceptable or even useful. However, for most other applications it is undesirable. In this paper, we present a collusion-free policy-based encryption primitive, called policy-based public-key encryption. We provide precise definition for the new primitive as well as for the related security model. Then, we describe a concrete implementation using pairings over elliptic curves and prove its security in the random oracle model.

Keywords: Pairing-Based Cryptography, Policy, Credentials.

1 Introduction

Policy-based encryption, recently formalized in [4], allows to encrypt a message with respect to a credential-based policy formalized as monotone boolean expression written in standard normal form. The encryption is so that only a user that is compliant with the policy is able to decrypt the message. A policy involves conjunctions (logical AND operation) and disjunctions (logical OR operation) of conditions, where each condition is fulfilled by a digital credential representing the signature of a specific credential issuer on a set of statements about a certain entity. A user is thus compliant with a policy if and only if he has been issued a qualified set of credentials for the policy i.e. a set of credentials fulfilling the combination of conditions defined by the policy. More generally, policy-based encryption belongs to an emerging family of cryptographic schemes sharing the ability to integrate encryption with credential-based access structures. This ability allows for several interesting applications in different contexts including but not restricted to oblivious access control [4,7,14], trust negotiation [6,9,13], and cryptographic workflow [3].

Suppose that Alice wants to send a sensitive message to Bob, while requiring that Bob fulfills a specific credential-based policy in order for him to be authorized to read

the message. In order to enforce her policy, Alice first encrypts her message according to her policy using a policy-based encryption algorithm, then she sends the resulting ciphertext to Bob. If Bob has access to a qualified set of credentials for Alice's policy, then he is compliant with the policy and can thus use his credentials to successfully decrypt the received message. An inherent property of the policy-based encryption primitive is that, in addition to Bob, any collusion of credential issuers or end users who are able to collect a qualified set of credentials for Alice's policy can decrypt the message as well. In some applications, where it is acceptable to assume that the credential issuers are trusted for not colluding with each other to spy the end user's communications and that no end user is willing to share his credentials with other end users, the collusion property is not a problem. In other applications, collusion of end users is useful when collaboration is required to authorize access to sensitive information, while collusion of credential issuers may even be desirable for law enforcement. However, for other applications such as trust establishment in large-scale open environments like the Internet, the collusion property is undesirable.

In this paper, we present a collusion-free variance of the policy-based encryption primitive defined in [4], that we call *policy-based public key encryption*. The intuition behind our encryption primitive is as follows: we assume that each end user is associated to a public/private key pair. We suppose that no end user is willing to share his private key with the others, and thus keeps secret his valuable key in a secure storage system such as a smart card. Furthermore, we suppose that a credential delivered by a credential issuer is associated to the requester's public key, and that it is issued after checking that the requester possesses the corresponding private key. As opposed to the basic policy-based encryption primitive, our encryption algorithm takes as input, in addition to a credential-based policy, the public key of the recipient the encrypted message is intended for. The policy taken as input is fulfilled by qualified sets of credentials for which all the credentials are associated to the recipient's public key. Our decryption algorithm is such that, in order to successfully decrypt the message, one needs to have access to a qualified set of credentials for the policy as well as to the recipient's private key. Thus, our policy-based public-key encryption primitive prevents collusions of credential issuers by making the decryption algorithm involve a secret element (private key) held only by the recipient the encrypted message is intended for. Besides, our primitive prevents collusions of end users by associating all the credentials fulfilling the policy according to which a message is encrypted to the same public key, and making these credentials useful only in conjunction with the corresponding private key.

In the following, we present the related work found so far in the literature.

1.1 Related Work

As said before, policy-based encryption and policy-based public-key encryption belong to an emerging family of cryptographic schemes sharing the ability to integrate encryption with credential-based access control structures. This ability is mainly enabled by pairings over elliptic curves, and more particularly by the Boneh-Franklin identity-based encryption from bilinear pairings [5]. Note that identity-based encryption could be seen as a particular case of policy-based encryption. In fact, an identity-based

encryption scheme corresponds to a policy-based encryption scheme for which policies are reduced to a single credential representing the signature of a centralized credential issuer (called private key generator) on the identity of an end user.

In [7], the authors present various applications of the use of multiple trusted authorities and multiple identities in the type of identity-based cryptography. They show how to perform encryption according to disjunctions and conjunctions of credentials. However, their solution remains restricted to a limited number of disjunctions. In [14], the author further pursues the ideas discussed in [7] and presents an elegant and efficient mechanism to perform access control based on encryption with respect to monotone boolean expressions written in standard normal forms. The proposed solution remains limited to credentials generated by a centralized trusted authority. Furthermore, it lacks adequate security arguments. In [4], the authors provide a further generalization of [14] by considering credentials that might be generated by independent credential issuers. They formalize the concept of policy-based cryptography and provide precise definitions for policy-based encryption and policy-based signature primitives. Furthermore, they show how such primitives could be used to enforce policies with respect to the data minimization principle according to which only strictly necessary information should be collected for a given purpose. Unfortunately, the presented schemes lack formal security analysis as for [14].

In [9], the authors introduce hidden credentials as a solution to perform privacy-enabled trust negotiation. Their solution uses the Boneh-Franklin encryption scheme [5] and relies on onion-like encryption and multiple encryption operations to deal, respectively, with conjunctions and disjunctions of credentials. Such approach remains inefficient in terms of both computational costs and bandwidth consumption (ciphertext size), especially when authorization structures become complex. In [6], the authors propose a solution to improve decryption efficiency as well as policy concealment when implementing hidden credentials with sensitive policies. They prove the chosen ciphertext security of their solution under the identity-based security models defined in [5].

Although used in application scenarios with different security requirements, the encryption schemes presented above share the fact that they allow to encrypt a message according to a credential-based policy so that only the users having access to a qualified set of credentials for the policy are able to successfully decrypt the message. While the schemes of [6,9] consider policies formalized as monotone boolean expressions written as general conjunctions and disjunctions of atomic terms, the schemes of [4,14] consider the ones written in standard normal forms. All the presented schemes are based on the Boneh-Franklin identity-based encryption primitive described in [5], from which they inherit the collusion property. In fact, the Boneh-Franklin scheme suffers from the key-escrow property i.e. the credential issuer is able to decrypt the confidential messages intended for the end users. As for the collusion property faced by policy-based encryption, the key-escrow property might be necessary in some contexts, especially within organizations, for monitoring and law enforcement. However, in most applications, it is undesirable.

In [1], the authors describe a modification of the Boneh-Franklin encryption scheme that allows to avoid the key-escrow problem. Their primitive, called certificateless

public-key encryption, requires each end user to have a public key. The encryption of a message is performed with respect to the identity of the recipient as well as with respect to his public key. The decryption algorithm requires both the recipient's private key and his identity credential. In [3], the authors consider general access structures and use a similar technique to achieve the policy-based encryption functionality while avoiding the collusion property. Their scheme could be seen as the collusion-free variance of the encryption scheme proposed in [6]. They underline the fact that their scheme supports cryptographic workflow, which is a feature inherited from the Boneh-Franklin encryption primitive and supported by the policy-based encryption primitive as well. They define formal security models to support their encryption primitive. Their 'recipient security model' considers indistinguishability against chosen plaintext attacks, where the adversary does not have access to the decryption oracle. Security against the stronger chosen ciphertext attacks is left as an open research problem.

1.2 Contributions and Outline of the Paper

In this paper, we define a new policy-based cryptographic primitive, called policy-based public-key encryption, and describe a provably secure concrete implementation based on bilinear pairings over elliptic curves. Our primitive allows to overcome the collusion problem faced by the original policy-based encryption primitive defined in [4]. We use a technique similar to the one used in [1] to overcome the key-escrow problem from which may suffer the identity-based encryption primitive defined in [5]. The escrow-free encryption scheme proposed in [3] may be considered as a policy-based public-key encryption scheme when applied to policies written in standard normal forms. Some may consider that restricting our scheme to standard normal forms is a limitation compared to the scheme of [3] which deals with general-form policies. We argue that this is not so, as in real-world scenarios security policies are typically written in standard normal forms. For example, the Web Service policy languages WS-Policy and WSPL consider policies converted to the standard disjunctive normal form. Our concrete scheme improves the performance of the key-escrow encryption scheme of [3] (when applied to standard-form policies) both in terms of computational cost and bandwidth consumption (size of the resulting ciphertext). Furthermore, we prove the security of our scheme against chosen ciphertext attacks as opposed to the approach of [3] that consider the weaker chosen plaintext attacks.

The rest of the paper is organized as follows: in Section 2, we first set the context for our encryption primitive including the terminology, the notation and the policy model. Then, we provide a precise definition for policy-based public-key encryption schemes as well as for the related security model. The latter adapts the strong security notion of indistinguishability against chosen ciphertext attacks to the specific features of the new policy-based cryptographic primitive. In Section 3, we describe a concrete policy-based public-key encryption primitive based on bilinear pairings over elliptic curves. Our scheme is a modification of the original policy-based encryption scheme described in [4] that integrates the public and private keys in the encryption and decryption algorithms respectively. As opposed to the scheme presented in [4] which lacks formal security analysis, we provide reductionist security arguments for our scheme in the random oracle model.

2 Definitions

2.1 Setting the Context

We consider a public key infrastructure where each end user holds a pair of keys (pk_u, sk_u) . An end user is identified by his public key pk_u . The public key does not have to be bound to the end user's name/identity (through public-key certification) as for standard PKI systems such as X.509. In fact, in large-scale open environments, the identity of an end user is rarely of interest to determining whether the end user could be trusted or authorized to conduct some sensitive transactions. Instead, statements about the end user such as attributes, properties, capabilities and/or privileges are more relevant. The validity of such statements is checked and certified by trusted entities called credential issuers through a digital signature procedure.

We consider a set of credential issuers $I = \{I_1, \dots, I_N\}$, where the public key of I_κ , for $\kappa \in \{1, \dots, N\}$, is denoted R_κ while the corresponding master key is denoted s_κ . We assume that a trustworthy value of the public key of each of the credential issuers is known by the end users. Any credential issuer $I_\kappa \in I$ may be asked by an end user to issue a credential corresponding to a set of statements. The requested credential is basically the digital signature of the credential issuer on an assertion denoted A^{pk_u} . The assertion contains, in addition to the set of statements, the end user's public key pk_u as well as a set of additional information such as the validity period of the credential. As the representation of assertions is out of the scope of this paper, they will simply be encoded as binary strings.

Upon receiving a request for generating a credential on assertion A^{pk_u} , a credential issuer I_κ first checks the fact that the requester has access to the private key sk_u associated to pk_u . Then, the credential issuer checks the validity of the assertion A^{pk_u} . If it is valid, then I_κ executes a credential generation algorithm and returns a credential denoted $\zeta(R_\kappa, A^{pk_u})$. Otherwise, I_κ returns an error message. Upon receiving the credential $\zeta(R_\kappa, A^{pk_u})$, the end user may check its integrity using I_κ 's public key R_κ . The process of checking the validity of a set of statements about a certain entity is out of the scope of this paper.

We consider credential-based policies formalized as monotone boolean expressions involving conjunctions (AND / \wedge) and disjunctions (OR / \vee) of credential-based conditions. A credential-based condition is defined through a pair $\langle I_\kappa, A^{pk_u} \rangle$ specifying an assertion $A^{pk_u} \in \{0, 1\}^*$ (about an end user whose public key is pk_u) and a credential issuer $I_\kappa \in I$ that is trusted to check and certify the validity of A^{pk_u} . An end user whose public key is pk_u fulfills the condition $\langle I_\kappa, A^{pk_u} \rangle$ if and only if he has been issued the credential $\zeta(R_\kappa, A^{pk_u})$.

We consider policies written in standard normal forms, i.e. written either in conjunctive normal form (CNF) or in disjunctive normal form (DNF). In order to address the two standard normal forms, we use the conjunctive-disjunctive normal form (CDNF) introduced in [14]. Thus, a policy denoted Pol^{pk_u} is written as follows:

$$Pol^{pk_u} = \wedge_{i=1}^m [\vee_{j=1}^{m_i} [\wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle]], \text{ where } I_{\kappa_{i,j,k}} \in I \text{ and } A_{i,j,k}^{pk_u} \in \{0, 1\}^*$$

Under the CDNF notation, policies written in CNF correspond to the case where $m_{i,j} = 1$ for all i, j , while policies written in DNF correspond to the case where $m = 1$.

Let $\zeta_{j_1, \dots, j_m}(Pol^{pk_u})$ denote the set of credentials $\{\{\zeta(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u})\}_{k=1}^{m_i, j_i}\}_{i=1}^m$, for some $\{j_i \in \{1, \dots, m_i\}\}_{i=1}^m$. Then, $\zeta_{j_1, \dots, j_m}(Pol^{pk_u})$ is a qualified set of credentials for policy Pol^{pk_u} .

2.2 Policy-Based Public-Key Encryption

A policy-based public-key encryption scheme (denoted in short PB-PKE) is specified by six algorithms: *System-Setup*, *Issuer-Setup*, *User-Setup*, *CredGen*, *Encrypt* and *Decrypt*, which we describe below.

System-Setup. On input of a security parameter k , this algorithm generates the public parameters \mathcal{P} which specify the different parameters, groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a public key space \mathcal{K} , a message space \mathcal{M} and a ciphertext space \mathcal{C} .

Issuer-Setup. This algorithm generates a random master key s_{κ} and the corresponding public key R_{κ} for credential issuer $I_{\kappa} \in I$.

User-Setup. This algorithm generates a random private key sk_u and the corresponding public key pk_u .

CredGen. On input of the public key R_{κ} of a credential issuer $I_{\kappa} \in I$ and an assertion $A^{pk_u} \in \{0, 1\}^*$, this algorithm returns the credential $\zeta(R_{\kappa}, A^{pk_u})$.

Encrypt. On input of a message $M \in \mathcal{M}$, a public key $pk_u \in \mathcal{K}$ and a policy Pol^{pk_u} , this algorithm returns a ciphertext $C \in \mathcal{C}$ representing the encryption of M with respect to policy Pol^{pk_u} and public key pk_u .

Decrypt. On input of a ciphertext $C \in \mathcal{C}$, a pair of keys (pk_u, sk_u) , a policy Pol^{pk_u} and a qualified set of credentials $\zeta_{j_1, \dots, j_m}(Pol^{pk_u})$, this algorithm returns either a message $M \in \mathcal{M}$ or \perp (for 'error').

The algorithms described above have to satisfy the following consistency constraint:

$$C = \text{Encrypt}(M, Pol^{pk_u}, pk_u) \Rightarrow \text{Decrypt}(C, Pol^{pk_u}, pk_u, sk_u, \zeta_{j_1, \dots, j_m}(Pol^{pk_u})) = M$$

Finally, we define $\phi_{j_1, \dots, j_m}(C, pk_u, Pol^{pk_u})$ to be the information from C that is required to correctly perform the decryption of C with respect to policy Pol^{pk_u} and public key pk_u using the qualified set of credentials $\zeta_{j_1, \dots, j_m}(Pol^{pk_u})$. A concrete example is given when describing our PB-PKE scheme. Such information is used in the specification of the security model associated to the policy-based public-key encryption primitive.

2.3 Security Model

Our security model for PB-PKE schemes follows the following reasoning: the standard acceptable notion of security for public key encryption schemes is indistinguishability against chosen ciphertext attacks (IND-CCA). Hence, it is natural to require that a PB-PKE scheme also satisfies this strong notion of security. However, the definition of this security notion must be adapted to the policy-based setting. A PB-PKE scheme is such that a user should not be able to decrypt a message if he does not fulfill the policy according to which the message was encrypted or if he does not have access to the

private key corresponding to the public key used to encrypt the message. Assume, for instance, that a user Alice wants to send a sensitive message to a user Bob whose public key is pk_b . Moreover, assume that Alice wants to be sure that Bob is compliant with a specific policy Pol^{pk_b} in order for Bob to be able to read the message. Thus, Alice uses a PB-PKE scheme to encrypt her message using Bob's public key pk_b according to her policy Pol^{pk_b} . Two attack scenarios should be considered:

- In the first scenario, a third user Charlie that has somehow access to a qualified set of credentials for policy Pol^{pk_b} tries to decrypt the intercepted message. For example, Charlie may represent a collusion of the different credential issuers specified by Pol^{pk_b} . As Charlie has not access to Bob's private key sk_b , he must not be able to successfully achieve the decryption. Because Charlie is not the legitimate recipient of the message he will be called *Outsider*.
- In the second scenario, the user Bob (who has access to the private key sk_b) does not have access to a qualified set of credentials for policy Pol^{pk_b} and tries to illegally decrypt the message. As Bob does not fulfill Alice's policy, he must not be able to successfully decrypt the message. As opposed to the Outsider adversary, Bob will be called *Insider*.

Our security model is defined in terms of an interactive game played between a challenger and an adversary, where the adversary can be either Insider or Outsider. The game consists of five stages: *Setup*, *Phase-1*, *Challenge*, *Phase-2* and *Guess*, which we describe below.

- **Setup.** On input of a security parameter k , the challenger does the following: (1) Run algorithm *System-Setup* to obtain the system public parameters \mathcal{P} which are given to the adversary, (2) Run algorithm *Issuer-Setup* once or multiple times to obtain a set of credential issuers $I = \{I_1, \dots, I_N\}$, (3) Run algorithm *User-Setup* to obtain a public/private key pair (pk_{ch}, sk_{ch}) . Depending on the type of the adversary, the challenger does the following: If the adversary is an Outsider, then the challenger gives to the adversary the public keys as well as the master keys of the credential issuers included in I . Furthermore, the challenger gives the public key pk_{ch} to the adversary while keeping secret the corresponding private key sk_{ch} . However, if the adversary is an Insider, then the challenger just gives to the adversary, in addition to the pair of keys (pk_{ch}, sk_{ch}) , the public keys of the credential issuers included in I while keeping secret the corresponding master keys.
- **Phase-1.** The adversary performs a polynomial number of oracle queries adaptively i.e. each query may depend on the replies to the previously performed queries.
- **Challenge.** This stage occurs when the adversary decides that the *Phase-1* stage is over. The adversary, be it Insider or Outsider, gives to the challenger two equal length messages M_0, M_1 and a policy $Pol_{ch}^{pk_{ch}}$ on which he wishes to be challenged. The challenger picks at random $b \in \{0, 1\}$, then runs algorithm *Encrypt* on input of the tuple $(M_b, pk_{ch}, Pol_{ch}^{pk_{ch}})$, and returns the resulting ciphertext C_{ch} to the adversary.
- **Phase-2.** The adversary performs again a polynomial number of adaptive oracle queries.
- **Guess.** The adversary outputs a guess b' , and wins the game if $b = b'$.

During the *Phase-1* and *Phase-2* stages, the adversary may perform queries to two oracles controlled by the challenger. On one hand, a credential generation oracle denoted **CredGen-O**. On the other hand, a decryption oracle denoted **Decrypt-O**. While the oracles are executed by the challenger, their input is specified by the adversary. The two oracles are defined as follows:

- **CredGen-O**. On input of a credential issuer $I_{\kappa} \in I$ and an assertion $A^{pk_u} \in \{0, 1\}^*$, run algorithm *CredGen* on input of the tuple (I_{κ}, A^{pk_u}) and return the resulting credential $\zeta(R_{\kappa}, A^{pk_u})$. Note that an Outsider does not need to perform queries to this oracle as he has access to the credential issuers' master keys. Besides, an Insider is not allowed to obtain a qualified set of credentials for the policy $Pol_{ch}^{pk_{ch}}$ which he is challenged on.
- **Decrypt-O**. On input of a ciphertext $C \in C$, a policy Pol^{pk_u} and a set of indices $\{j_1, \dots, j_m\}$, first run algorithm *CredGen* multiple times to obtain the qualified set of credentials $\zeta_{j_1, \dots, j_m}(Pol^{pk_{ch}})$, then run algorithm *Decrypt* on input of the tuple $(C, pk_{ch}, sk_{ch}, Pol^{pk_{ch}}, \zeta_{j_1, \dots, j_m}(Pol^{pk_{ch}}))$, and return the resulting output. Note that an adversary, be it Insider or Outsider, cannot perform a query to oracle *Decrypt-O* on a tuple $(C, Pol_{ch}^{pk_{ch}}, \{j_1, \dots, j_m\})$ such that $\varphi_{j_1, \dots, j_m}(C, pk_{ch}, Pol_{ch}^{pk_{ch}}) = \varphi_{j_1, \dots, j_m}(C_{ch}, pk_{ch}, Pol_{ch}^{pk_{ch}})$.

The game described above is denoted $IND\text{-}Pol\text{-}CCA_{PK}^X$, where $X = I$ for Insider adversaries and $X = O$ for Outsider adversaries. A formal definition of chosen ciphertext security for PB-PKE schemes is given below. As usual, a real function g is said to be negligible if $g(k) \leq \frac{1}{f(k)}$ for any polynomial f .

Definition 1. *The advantage of an adversary \mathcal{A}^X in the $IND\text{-}Pol\text{-}CCA_{PK}^X$ game is defined to be the quantity $Adv_{\mathcal{A}^X} = |\Pr[b = b'] - \frac{1}{2}|$. A PB-PKE scheme is $IND\text{-}Pol\text{-}CCA_{PK}^X$ secure if no probabilistic polynomial time adversary has a non-negligible advantage in the $IND\text{-}Pol\text{-}CCA_{PK}^X$ game.*

Note. Our security model could be viewed as an extension to the policy-based public-key setting of the $IND\text{-}ID\text{-}CCA$ model defined in [5]. In $IND\text{-}ID\text{-}CCA$, the adversary is not allowed to make decryption queries on the challenge tuple (C_{ch}, ID_{ch}) . In the policy-based public-key setting, for an encrypted message with respect to a policy with disjunctions, there is more than one possible qualified set of credentials that can be used to perform the decryption. That is, forbidding the adversary from making decryption queries on the challenge tuple $(C_{ch}, Pol_{ch}^{pk_{ch}})$ is not sufficient anymore. In fact, we may have tuples such that $(C, Pol^{pk_{ch}}) \neq (C_{ch}, Pol_{ch}^{pk_{ch}})$ while $\varphi_{j_1, \dots, j_m}(C, Pol^{pk_{ch}}) = \varphi_{j_1, \dots, j_m}(C_{ch}, Pol_{ch}^{pk_{ch}})$. Decryption queries on such tuples should then be forbidden. \diamond

3 Our PB-PKE Scheme

3.1 Description

Before describing our PB-PKE scheme, we define algorithm *BDH-Setup* as follows:

BDH-Setup. Given a security parameter k , generate a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, e, P)$ where the map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear pairing, $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, *)$ are two groups of the

same order q , and P is a random generator of \mathbb{G}_1 . The generated parameters are such that the Bilinear Diffie-Hellman Problem (denoted BDHP) is hard.

Note-1. We recall that a bilinear pairing satisfies the following three properties: (1) Bilinear: for $Q, Q' \in \mathbb{G}_1$ and for $a, b \in \mathbb{Z}_q^*$, $e(a \cdot Q, b \cdot Q') = e(Q, Q')^{ab}$, (2) Non-degenerate: $e(P, P) \neq 1$ and therefore it is a generator of \mathbb{G}_2 , (3) Computable: there exists an efficient algorithm to compute $e(Q, Q')$ for all $Q, Q' \in \mathbb{G}_1$. \diamond

Note-2. BDHP is defined as follows: on input of a tuple $(P, a \cdot P, b \cdot P, c \cdot P)$ for randomly chosen $a, b, c \in \mathbb{Z}_q^*$, compute the value $e(P, P)^{abc}$. The hardness of BDHP can be ensured by choosing groups on supersingular elliptic curves or hyperelliptic curves over finite fields and deriving the bilinear pairings from Weil or Tate pairings. The hardness of BDHP implies the hardness of the so called Computational Diffie-Hellman Problem (denoted CDHP) which is defined as follows: on input of a tuple $(P, a \cdot P, b \cdot P)$ for randomly chosen $a, b \in \mathbb{Z}_q^*$, compute the value $ab \cdot P$. As we merely apply these mathematical primitives in this paper, we refer for instance to [10,15] for more details. \diamond

Our PB-PKE scheme consists of the algorithms described below.

System-Setup. On input of a security parameter k , do the following:

1. Run algorithm *BDH-Setup* to obtain a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, e, P)$
2. Let $\mathcal{M} = \{0, 1\}^n$, $\mathcal{K} = \mathbb{G}_1$ and $\mathcal{C} = \mathbb{G}_1 \times (\{0, 1\}^n)^* \times \{0, 1\}^n$ (for some $n \in \mathbb{N}^*$)
3. Define four hash functions: $H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1$, $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$,
 $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^n$ and $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^n$
4. Let $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_2, e, P, n, H_0, H_1, H_2, H_3)$.

Issuer-Setup. Let $I = \{I_1, \dots, I_N\}$ be a set of credential issuers. Each credential issuer $I_{\kappa} \in I$ picks at random a secret master key $s_{\kappa} \in \mathbb{Z}_q^*$ and publishes the corresponding public key $R_{\kappa} = s_{\kappa} \cdot P$.

User-Setup. This algorithm picks at random a private key $sk_u \in \mathbb{Z}_q^*$ and computes the corresponding public key $pk_u = sk_u \cdot P$.

CredGen. On input of issuer $I_{\kappa} \in I$ and assertion $A^{pk_u} \in \{0, 1\}^*$, this algorithm outputs $\zeta(R_{\kappa}, A^{pk_u}) = s_{\kappa} \cdot H_0(A^{pk_u})$.

Encrypt. On input of message $M \in \mathcal{M}$, public key pk_u and policy Pol^{pk_u} , do the following:

1. Pick at random $t_i \in \{0, 1\}^n$ (for $i = 1, \dots, m$)
2. Compute $r = H_1(M \| t_1 \| \dots \| t_m)$, then compute $U = r \cdot P$ and $K = r \cdot pk_u$
3. Compute $\pi_{i,j} = \prod_{k=1}^{m_{i,j}} e(R_{\kappa_{i,j,k}}, H_0(A_{i,j,k}^{pk_u}))$ (for $j = 1, \dots, m_i$ and $i = 1, \dots, m$)
4. Compute $\mu_{i,j} = H_2(K \| \pi_{i,j}^r \| i \| j)$, then $v_{i,j} = t_i \oplus \mu_{i,j}$ (for $j = 1, \dots, m_i$ and $i = 1, \dots, m$)
5. Compute $W = M \oplus H_3(t_1 \| \dots \| t_m)$
6. Return $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^m, W)$

The intuition behind the encryption algorithm is as follows: each conjunction of conditions $\bigwedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle$ is first associated to a mask $\mu_{i,j}$ that depends not only on the different credentials related to the specified conditions but also on the specified public key. Then, for each index $i \in \{1, \dots, m\}$, a randomly chosen intermediate key t_i is associated to the disjunction $\bigvee_{j=1}^{m_i} \bigwedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle$. Finally, each intermediate key t_i is encrypted

m_i times using each of the masks $\mu_{i,j}$. This way, it is sufficient to compute any one of the masks $\mu_{i,j}$ in order to be able to retrieve t_i . In order to be able to retrieve the encrypted message, an entity needs to retrieve all the intermediate keys t_i using not only a qualified set of credentials for policy Pol^{pk_u} , but also the private key sk_u corresponding to pk_u .

Decrypt. On input of ciphertext $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^m, W)$, the pair of keys (pk_u, sk_u) , policy Pol^{pk_u} and the qualified set of credentials $\zeta_{j_1, \dots, j_m}(Pol^{pk_u}, pk_u)$, do the following:

1. Compute $\tilde{\pi}_{i,j_i} = e(U, \sum_{k=1}^{m_{i,j_i}} \zeta(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u}))$ (for $i = 1, \dots, m$), then compute $\tilde{K} = sk_u \cdot U$
2. Compute $\tilde{\mu}_{i,j_i} = H_2(\tilde{K} \parallel \tilde{\pi}_{i,j_i} \parallel i \parallel j_i)$, then compute $t_i = v_{i,j_i} \oplus \tilde{\mu}_{i,j_i}$ (for $i = 1, \dots, m$)
3. Compute $M = W \oplus H_3(t_1 \parallel \dots \parallel t_m)$, then compute $r = H_1(M \parallel t_1 \parallel \dots \parallel t_m)$
4. If $U = r \cdot P$, then return the message M , otherwise return \perp

Note. Our PB-PKE scheme is such that $\varphi_{j_1, \dots, j_m}(C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^m, W), Pol^{pk_u})$ consists of the values U and W as well as the pairs $\{(v_{i,j_i}, \wedge_{k=1}^{m_{i,j_i}} (I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u}))\}_{i=1}^m$. \diamond

3.2 Consistency and Efficiency

The algorithms described above satisfy the standard consistency constraint. In fact, we have, on one hand, $\tilde{K} = sk_u \cdot U = sk_u \cdot (r \cdot P) = r \cdot (sk_u \cdot P) = r \cdot pk_u$. On the other hand, the following holds

$$\tilde{\pi}_{i,j_i} = e(r \cdot P, \sum_{k=1}^{m_{i,j_i}} s_{\kappa_{i,j_i,k}} \cdot H_0(A_{i,j_i,k}^{pk_u})) = \prod_{k=1}^{m_{i,j_i}} e(s_{\kappa_{i,j_i,k}} \cdot P, H_0(A_{i,j_i,k}^{pk_u}))^r = \pi_{i,j_i}^r$$

The essential operation in pairing-based cryptography is pairing computations. Although such operation can be optimized, it still have to be minimized. In Table 1, we provide the computational costs of our encryption and decryption algorithms in terms of pairing computations as well as the size of the resulting ciphertext. Note that l_1 denotes the bit-length of the bilinear representation of an element of group \mathbb{G}_1 .

Table 1. Performance of our PB-PKE scheme compared with the scheme of [3]

	Encryption	Decryption	Ciphertext Size
Our PB-PKE scheme	$\sum_{i=1}^m \sum_{j=1}^{m_i} m_{i,j}$	m	$l_1 + (\sum_{i=1}^m m_i) \cdot n + n$
The scheme of [3]	$\sum_{i=1}^m \sum_{j=1}^{m_i} m_{i,j}$	$\sum_{i=1}^m m_{i,j_i}$	$l_1 + (\sum_{i=1}^m \sum_{j=1}^{m_i} m_{i,j}) \cdot n + n$

In Table 1, we provide the performance of the key-escrow scheme of [3] when applied to policies written in standard normal forms following the notation defined in Section 2. While the encryption algorithms require the same amount of pairing computations, our decryption algorithm more efficient as $m_{i,j_i} \geq 1$ for $i = 1, \dots, m$. Furthermore, as $m_{i,j} \geq 1$ for $j = 1, \dots, m_i$ and $i = 1, \dots, m$, the size of the ciphertexts resulting from our scheme is at least as short as the one of the ciphertexts produced by the scheme of [3].

Note. As for standard asymmetric encryption schemes, PB-PKE schemes are much less efficient than symmetric encryption schemes. In practice, they should be used to exchange the symmetric (session) keys that are used for bulk encryption. \diamond

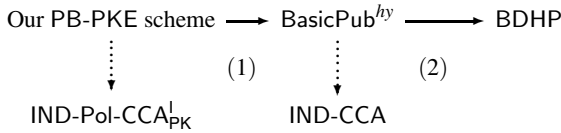
3.3 Security

In the following, we show respectively that our PB-PKE scheme is both $\text{IND-Pol-CCA}_{PK}^I$ and $\text{IND-Pol-CCA}_{PK}^O$ secure in the random oracle model.

Notation. Given the notation used in Section 2, the maximum values that the quantities m , m_i and $m_{i,j}$ can take are denoted, respectively, $m_{\vee\wedge} \geq 1$, $m_{\vee} \geq 1$ and $m_{\wedge} \geq 1$. We assume that these upper-bounds are specified during system setup. \diamond

Theorem 1. *Our PB-PKE scheme is $\text{IND-Pol-CCA}_{PK}^I$ secure in the random oracle model under the assumption that BDHP is hard.*

Proof. Theorem 1 follows from a sequence of reduction arguments that are summarized in the following diagram:



1. Lemma 1 shows that an $\text{IND-Pol-CCA}_{PK}^I$ attack on our PB-PKE scheme can be converted into an IND-CCA attack on the BasicPub^{hy} algorithm defined in [5].
2. In [5], algorithm BasicPub^{hy} is shown to be IND-CCA secure in the random oracle model under the assumption that BDHP is hard.

Lemma 1. *Let \mathcal{A}° be an $\text{IND-Pol-CCA}_{PK}^I$ adversary with advantage $\text{Adv}_{\mathcal{A}^\circ} \geq \epsilon$ when attacking our PB-PKE scheme. Assume that \mathcal{A}° has running time $t_{\mathcal{A}^\circ}$ and makes at most q_c queries to oracle CredGen-O , q_d queries to oracle Decrypt-O as well as q_0 queries to oracle H_0 . Then, there exists an IND-CCA adversary \mathcal{A}^\bullet the advantage of which, when attacking the BasicPub^{hy} scheme, is such that $\text{Adv}_{\mathcal{A}^\bullet} \geq F(q_c, q_d, q_0, N, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) \cdot \epsilon$. Its running time is $t_{\mathcal{A}^\bullet} = O(t_{\mathcal{A}^\circ})$.*

Note-1. Lemma 1 stated below uses the quantity $\zeta = F(q_c, q_d, q_0, N, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ defined as follows:

$$\zeta = \left(1 - \frac{q_c m_{\vee\wedge} m_{\vee}}{N q_0}\right) \cdot \left(1 - \frac{q_d \Upsilon'(N q_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})}{\Upsilon(N q_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})}\right) \cdot \frac{1}{\Upsilon(N q_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})}$$

where

$$\Upsilon'(N q_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) = \Upsilon(N q_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) - \Upsilon(N q_0 - (m_{\vee\wedge} m_{\vee})^2, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) - 1$$

Computing $F(\cdot)$ relies on computing the quantity $\Upsilon(X, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$, which is defined to be the total number of 'minimal' (reduced) policies written in CDNF, given the upper-bounds $(m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ and X possible credential-based conditions. Computing $\Upsilon(X, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ is similar, but not exactly the same as the problems of computing the number of monotone boolean functions of n variables (Dedekind's Problem [12])

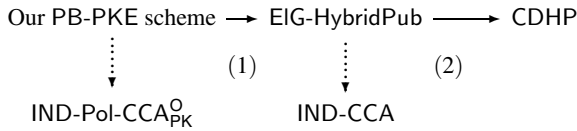
and computing the number of antichains on a set $\{1, \dots, n\}$ [11]. As opposed to these problems, the order of the terms must be taken into consideration when dealing with our policies. This is a typical, yet interesting, 'counting' problem. Due to space limitation, we do not elaborate more on the details. \diamond

Note-2. In the case where $N = m_{\vee \wedge} = m_{\vee} = m_{\wedge} = 1$, we have $\Upsilon'(Nq_0, m_{\vee \wedge}, m_{\vee}, m_{\wedge}) = 0$ and $\Upsilon(Nq_0, m_{\vee \wedge}, m_{\vee}, m_{\wedge}) = q_0$. In this case, our PB-PKE scheme when attacked by the Insider adversary is equivalent to the FullIdent scheme of [5]. Note that our results match Result 5 of [8]. In fact, our reductionist security proof follows a strategy similar to the one used in [8]. \diamond

Note-3. The result of our security reduction remains theoretical. The function $F(\cdot)$ depends exponentially on the policy size bounds which is not acceptable in practice. We are currently working on improving the tightness of our reduction in order to determine exact security arguments for real-world scenarios. \diamond

Theorem 2. *Our PB-PKE scheme is $\text{IND-Pol-CCA}_{\text{PK}}^{\text{O}}$ secure in the random oracle model under the assumption that CDHP is hard.*

Proof. Theorem 2 follows from two reduction arguments that are summarized in the following diagram:



1. Lemma 2 shows that an $\text{IND-Pol-CCA}_{\text{PK}}^{\text{O}}$ attack on our PB-PKE scheme can be converted into an IND-CCA attack on the EIG-HybridPub algorithm defined in [2].
2. In [2], algorithm $\text{EIG-HybridPub}^{\text{hy}}$ is shown to be IND-CCA secure in the random oracle model under the assumption that CDHP is hard.

Lemma 2. *Let \mathcal{A}° be an $\text{IND-Pol-CCA}_{\text{PK}}^{\text{O}}$ adversary with advantage $\text{Adv}_{\mathcal{A}^{\circ}} \geq \epsilon$ when attacking our PB-PKE scheme. Then, there exists an IND-CCA adversary \mathcal{A}^{\bullet} the advantage of which, when attacking the EIG-HybridPub scheme, is such that $\text{Adv}_{\mathcal{A}^{\bullet}} \geq \epsilon$. Its running time is $t_{\mathcal{A}^{\bullet}} = O(t_{\mathcal{A}^{\circ}})$.*

The details of the proofs of Lemma 1 and Lemma 2 are given in the full version of this paper.

4 Conclusion

In this paper, we presented a collusion-free policy-based encryption primitive. We provided formal definitions for the new primitive and described a concrete implementation using bilinear pairings over elliptic curves. We defined a strong security model for our primitive following the notion of indistinguishability against chosen ciphertext attacks, and proved the security of our pairing-based scheme in the random oracle model. The

goal of the new primitive is to overcome the weakness of the original policy-based encryption primitive defined in [4] when used in application scenarios for which collusions of credential issuers and end users are undesirable. The key-escrow encryption scheme presented in [3] allows to achieve the same security goals when applied to policies written in standard normal forms. Our proposal improves the scheme of [3] in terms of both performance and formal security analysis. Our security analysis remains theoretical and the results of our reductionist proof are unacceptable for practical use of our primitive. We are currently working on improving the tightness of our reduction and determining exact security parameters for real-world scenarios. A target application for our primitive is trust establishment and negotiation in large-scale open environments.

References

1. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, pages 452–473. Springer-Verlag, 2003.
2. S.S. Al-Riyami. Cryptographic schemes based on elliptic curve pairings. Ph.D. Thesis, Royal Holloway, University of London, 2004.
3. S.S. Al-Riyami, J. Malone-Lee, and N.P. Smart. Escrow-free encryption supporting cryptographic workflow. Cryptology ePrint Archive, Report 2004/258, 2004. <http://eprint.iacr.org/>.
4. W. Bagga and R. Molva. Policy-based cryptography and applications. In *Proceedings of Financial Cryptography and Data Security (FC'05)*, volume 3570 of *LNCS*, pages 72–87. Springer-Verlag, 2005.
5. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.
6. R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. Cryptology ePrint Archive, Report 2004/109, 2004. <http://eprint.iacr.org/>.
7. L. Chen, K. Harrison, D. Soldera, and N. Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *Proceedings of the International Conference on Infrastructure Security*, pages 260–275. Springer-Verlag, 2002.
8. D. Galindo. Boneh-franklin identity based encryption revisited. To appear in *Proceedings of 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*.
9. J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society*. ACM Press, 2003.
10. A. Joux. The weil and tate pairings as building blocks for public key cryptosystems. In *Proceedings of the 5th International Symposium on Algorithmic Number Theory*, pages 20–32. Springer-Verlag, 2002.
11. J. Kahn. Entropy, independent sets and antichains: a new approach to dedekind's problem. In *Proc. Amer. Math. Soc.* 130, pages 371–378, 2002.
12. D. Kleitman. On dedekind's problem: the number of monotone boolean functions. In *Proc. Amer. Math. Soc.* 21, pages 677–682, 1969.
13. N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *Proceedings of the 22nd annual symposium on Principles of distributed computing*, pages 182–189. ACM Press, 2003.
14. N. Smart. Access control using pairing based cryptography. In *Proceedings CT-RSA 2003*, pages 111–121. Springer-Verlag LNCS 2612, April 2003.
15. Y. Yacobi. A note on the bilinear diffie-hellman assumption. Cryptology ePrint Archive, Report 2002/113, 2002. <http://eprint.iacr.org/>.

Using Multiple Smart Cards for Signing Messages at Malicious Terminals

István Zsolt Berta

Microsec Ltd.

Abstract. Having no trusted user interface, smart cards are unable to communicate with the user directly. Communication is possible with the aid of a terminal only, which leads to several security problems. For example, if the terminal is untrusted (which is a very typical scenario), it may perform a man-in-the-middle attack. Thus, a malicious terminal can make the user sign documents that she would not sign otherwise. A signature that a card computes at a malicious terminal does not prove anything about the content of the signed document. What it does prove, is that the user did insert her card into a malicious terminal and she did intend to sign – something.

In this paper we propose a solution where a user has multiple smart cards, and each card represents a 'signal', a certain piece of information. The user encodes her message by using a subset of her cards for signing at the untrusted terminal. The recipient decodes the message by checking which cards were used. We also make use of time stamps from a trusted time stamping authority to allow cards to be used more than once.

1 Introduction

Electronic commerce applications require participants to send messages over a network. If these messages contain sensitive information, they need to be protected. For instance, a remote recipient who makes important decisions based on such a message would like to be convinced that the message is authentic: it originates from the sender and it has not been modified underway.

Digital signatures provide a way for a sender to ensure the authenticity of the message. Moreover, signatures allow a recipient to later prove it to a third party that the message originates from the sender. Today some signatures are even legally binding, so courts accept them as non-repudiable evidence.

In this paper that practical scenario is considered, where a human user would like to send a digitally signed message to a remote partner. Our user is mobile, and she does not have a trusted computer on her. All she has is a smart card that stores her private signing key. She would like to send a message of utmost importance, and she supposes that every terminal she can access is possibly malicious.

Signing messages at malicious terminals is dangerous. The digital signature is a very complex operation, the user is unlikely to be able to compute it without any computational aid. Typically, the signature is either computed by a terminal, or by the smart card of the user. It is unwise to trust a malicious terminal with computing a digital signature, because it may abuse the user's signing key. If the signature is computed by the smart

card of the user, the terminal cannot get hold of the key itself. However, the smart card does not have any user interface of its own, so the user still has to rely on the malicious terminal for sending the message to the smart card before the card signs the message. In this step, the malicious terminal may perform an obvious attack: it may replace the message with another one that the user would not want to sign.

As a matter of fact, malicious terminal can make the user sign an arbitrary message.

2 Related Work

The problem of man-in-the-middle attacks of untrusted terminals was addressed by Abadi et al. first, by analyzing the dangers of delegation of rights to a terminal. [1] They show that this problem could be solved with a smart card that has peripherals to communicate directly with the user, and they also show secure protocols for such a device. Later on, they strip as much of these peripherals from the card, as possible. Schneier and Shostack also give a good overview of this problem. [2] Literature provide three branches of solutions for the problem of sending authentic messages from untrusted terminals. Some works (e.g. [3], [4], [5]) propose solutions based on *super smart cards*, that do have some peripherals for communicating with the user directly.

Some other works are based on *human-computer cryptography*, they provide solutions where the human user protects the message without the help of a smart card. Examples for this approach are visual cryptography, and the human authentication scheme of Matsumoto. [6], [7] These solutions rely on the fact the human can perform certain special operation much faster than computer can. There are some cryptographic algorithms that are optimized for being executed by humans. The “Solitaire” encryption algorithm is a good example for this. [8] We do not know of such an algorithm for message authentication.

In contrast to the above two branches, this paper provides a solution based on *realistic smart cards*, devices that exist today and that are feasible to deploy at large scale. In our solution the human user does not have to perform any cryptographic operations for authenticating the message.

In case of solutions based on realistic smart cards, it is assumed that the card does not have any peripherals for direct communication with the user. This means that the user,

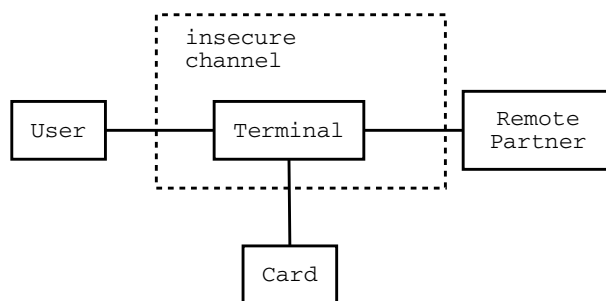


Fig. 1. A model for systems with malicious terminals

the card and the remote partner of the user are three entities that can communicate with each other only through the malicious terminal that is a part of the insecure channel. (See Figure 1.) Thus, establishing secure communication between the user and her card is exactly the same problem as establishing secure communication between the user and the remote partner.

The model on Figure 1 was introduced by Stabell-Kulo et al. [9] who proposed that the user may authenticate her message by encrypting it using a one-time-pad and a monoalphabetic substitution table. Unfortunately, there are severe key management problems concerning the one-time-pad. For example, in case of long messages the user is unable to memorize the long one-time keys.

Asokan et al. proposed a solution that allows the user to authenticate untrusted terminals, but their solution does not deal with the problem of sending messages from malicious terminals. [10]

Berta and Vajda provided a formal model for the computational abilities a human user, and they have formally proven that this problem has no solution in case of messages that are longer than the key the user can memorize (which implies that one-time-pads cannot be used securely). They have proven that if the human user is unable to securely encrypt or authenticate a message without the help of a trusted computer, than the human is unable to take part in any protocol that would allow her to establish an encrypted or authenticated communications channel that the malicious terminal cannot easily attack. [11] This implies that the problem of sending authentic messages from malicious terminals cannot be solved if we model the user as a slow computational network device. If there is a solution to this problem, it should be sought in a different model.

Berta et al. proposed a solution that change the model by posing lesser requirements on digital signatures computed at untrusted terminals. [12], [13] In this solution, the user can later revoke unintended signatures under well-defined circumstances. This solution is suitable for signing transactions of small value, but fails if the user can gain too much by repudiating a single signature. In contrast to the above work, signatures cannot be revoked in the solution we propose in this paper, so they can be used for authenticating transactions of arbitrary high value.

The solutions of Gruschka et al. ([14]) and Girard et al. ([15]) also pose lesser requirements on digital signature: in their model signatures cannot be used for any purpose. Their solutions protect the user by making the trusted smart card enforce limitations on what documents the user can sign. Thus if the card limits the messages that can be signed to small value bank transfers, the malicious terminal cannot make it compute a signature over a million-dollar contract. There are two problems with this approach: The first one is that this solution prevents the user from having a general purpose digital signature. The second one is that the malicious terminal can still alter the message within the limitations posed by the smart card. For example, if the card limits the messages that can be signed to million-dollar contracts, then the user would like to be absolutely sure, which million-dollar contract she signs. The work of Gruschka et al. also proposes solutions where certain parts of the terminal are secure, but in this paper we consider that particular situation only where the terminal is fully under the control of the attacker. The solution proposed in this paper does not place any restrictions on the messages the user can authenticate.

Berta and Vajda also proposed a solution [16] where the human user does not sign a plaintext message, but she signs a biometric (e.g. voice or video) message with her trusted smart card. The biometric message carries her biometric identity, so if the malicious terminal would like to make her sign a different message, then the malicious terminal needs to counterfeit the biometry of the user too. We assume that it takes significantly more time to counterfeit a biometric message than to counterfeit a plaintext one. There is a timestamping mechanism in the system that ensures that the malicious terminal has very little time to tamper with the biometric message before it is signed by the trusted smart card of the user. The solution we propose in this paper does not rely on the fuzziness of biometry, it employs purely algorithmic countermeasures.

3 Model

We can summarize previous solutions with the following statement: *A human user, who does not have a trusted computational device at her disposal, is helpless.* If the terminal she is using is malicious, it can make the user sign an arbitrary message. Previous works either assume that the user has some trusted computational device she can communicate with through a trusted channel, or they assume that the user has an exceptional memory or some exceptional computational abilities – which means that she (i.e. her brain) is the aforementioned trusted computational device. Some works try to protect the user by placing limitations on what documents she can sign, and prevent her from authenticating any message she chooses. Some others pose different requirements on signatures (and e.g. allow them to be revoked).

We do not know of any solution that allows the average human user to authenticate long messages without having a trusted computational device at her disposal that she can communicate with through a trusted channel.

In this paper, we propose such a solution. Our solution does not require the user to memorize any keybit or to perform any computations. Our solution includes trusted computational devices (smart cards), but does not assume that the user can communicate with them in a trusted way. What the user has to do, is allowing and blocking communication between the untrusted terminal and the smart cards. Naturally, the 'work' she has to perform is directly proportional to the length of her message. Our solution requires a global trusted time stamping service to be available.

According to the model we use in the rest of this paper, user U is a human who would like to send messages to remote partner, recipient R from malicious terminal T . The user has one or more smart cards that she can use for signing messages, and each smart card contains one signing key.

We define our model by the following assumptions:

1. The user, her smart cards and the remote partner can communicate with each other through the malicious terminal only (see Figure 1).
2. The terminal is fully under the control of the attacker, so the attacker is able to record, modify and replay any message passing through the untrusted terminal.
3. There is no cryptographic algorithm that a human user can execute to protect the authenticity of messages from the malicious terminal. This means that existing cryptographic algorithms are either too complex, so a human user cannot execute them

on “long” messages (i.e. where one-time-pads are out of the question), or they are too weak, so a malicious terminal can easily break them. (See [11] for the formalization of this assumption.)

4. The user is able to block the communication channels between her smart cards and the malicious terminal. She has a straightforward way for doing this: the terminal is not able to communicate with a card unless the user inserts it into the card reader of the terminal.

On the other hand, if the user inserts a card into card reader of the terminal, the terminal can make the card compute one or more¹ signatures over messages chosen by the malicious terminal until the card is removed from the reader. (We do not make any assumption on whether the card protects the signing keys by PIN codes. Although PIN codes are useful against e.g. card theft, they provide little protection against the threat of untrusted terminals, so their use is not discussed in this paper.)

5. The attacker cannot attack the digital signature algorithm with a non-negligible probability. This means that the attacker has a negligible chance to produce the signature for a given datablock, if the attacker does not know the corresponding private key. The attacker also has just a negligible chance for obtaining the signing key from a signature.
6. Smart cards generate their own signing keys, and it is impossible to extract these keys from the smart cards.

We assume that the attacker controls the terminal of the user when she sends messages m_1, m_2, \dots, m_n . The aim of the attacker is to make the remote partner accept message m' as an authentic message originating from the user, where m' is not a message originating from the user ($\forall i : m' \neq m_i$). [17]

4 Our Solution in a Nutshell

According to Assumption 3, the user is unable to perform cryptographic operations that would allow her to send authentic messages to her smart card. This implies that the malicious terminal can make the user sign an arbitrary message, so in case of malicious terminals, *the digital signature of the card does not prove anything about the content of the signed message*.

However, a signature computed at a malicious terminal is not totally useless. Such a signature can still be used for proving the following: [16]

- The user signed *something* with her card.
- Whatever datablock was signed, it was not altered after the signing.
- If there are any securely obtained timestamps protecting the signature, they can be used for proving the time of the signing.

¹ It is possible to prevent the terminal from obtaining a signature without the user entering the PIN code by using trusted smart card readers with integrated PIN pads. However, not even a trusted reader with an integrated PIN pad can prevent the terminal from replacing the message the user would like to sign with an arbitrary message. In this paper we assume that the terminal is completely under the control of the attacker and it does not have any parts that are trusted by the user (Assumption 2).

In the solution we propose in this paper, we make heavy use of the first and the last of the above statements.

Let us assume that if the user wishes to send message '1' from a malicious terminal, and she inserts her card and signs a message. If the remote partner receives any message signed with the user's private key, then the remote partner can make sure that the user sent message '1'. If the remote partner does not receive any message signed by the user's private key, then the remote partner cannot decide whether the user did not send anything, or she sent message '1' but the malicious terminal blocked it.

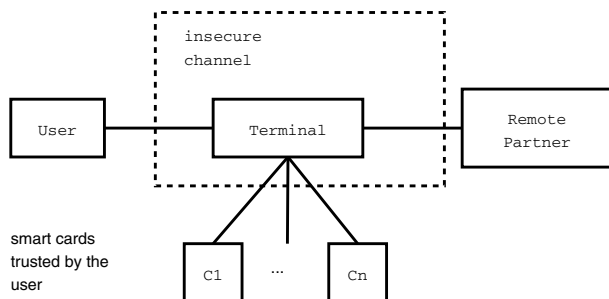


Fig. 2. Our model with multiple trusted smart cards

There are two problems with the above approach. On the one hand, the signing key (and the corresponding smart card) can be used only once. The user should not use the card ever again, otherwise the remote partner may not be able to differentiate between the received messages. On the other hand, it is not obvious how to send message '0'.

We propose that if the user has multiple smart cards, she can authenticate messages even at malicious terminals. In the solution we propose, the user does not authenticate the message by signing it with her card, but by signing some datablocks with a subset of her cards. (Figure 2 shows the model with the user having multiple trusted smart cards.) Henceforth, we call a datablock and a signature on it created with one of the user's cards a *signal*. Signals should have a structure that can be recognized by the remote partner, but the content of the signed datablock in the signal is irrelevant. Each card of the user contains a different private key, so each card produces a different signal. At the untrusted terminal, *the human user encodes her message as a list of signals*, and the remote recipient obtains the message by decoding the list of signals received from the user. As the malicious terminal is able to obtain signatures on any datablock, we assume that the digital certificates of the signing keys clearly state that signatures with the signing keys can be used for sending signals only, and such signatures do not prove the consent of the user.

There is a theoretically workable solution for sending an n -bit-long message from an untrusted terminal: The user should be equipped with $2n$ smart cards, cards *one*[1.. n] for sending signal '1', and cards *zero*[0.. n] for sending signal '0'. If the user wishes to send message '101', she has to send three signals, i.e. she has to perform three signatures with three cards: one signature with card *one*[1], another one with card *zero*[2]

and one with card *one*[3]. The remote partner expects three signals to arrive, and each signal to denote a bit at a different position in the message. Unfortunately, this solution allows the user to send only one message, and this message must have a fixed length. If n , the length of the message is large, then this solution requires the user to carry a truckload of cards on her – cards she cannot use ever again. We improve this solution in the next section by allowing the user to use cards more than once. More precisely, we allow her to send multiple messages of variable length while carrying a constant number of cards only.

5 A Detailed Practical Solution

It is possible to reduce the number of cards required for sending the message by introducing the notion of time in our protocol. If the time of signing was included in the signals, the remote partner would be able to detect if the malicious terminal tampered with the order of the signals before sending it to the remote partner.

We can safely assume that the user has a source of time independent from the terminal, e.g. a watch. (The biometric protocol of Berta and Vajda also uses this concept. [16]) According to Assumptions 1 and 3, the user cannot send the value of the exact time to the smart card in an authentic way. Unfortunately, most smart cards today do not have a timer of their own, so they need to rely on a trusted source of time, like a time stamping authority (TSA). A time stamping authority is a trusted party who puts digitally signed timestamps on incoming messages. A TSA who receives input x answers $timestamp(x) = (t_{current}, sign_{TSA}(t_{current}, x))$. The user trusts TSA for having a secure source of time and for functioning correctly. The TSA also handles its signing key in a secure way (i.e. Assumption 6 is true for the TSA).

We assume the user and the remote partner agreed on a set of time frames. The first frame is between t_1 and t_2 , the second frame is between t_2 and t_3 , etc. We assume that the time frames are of equal length, i.e. $\forall i, t_{i+1} - t_i = t_s$.

It should not be complicated to agree on these time frames and the time frames should not be kept secret. For example, the user and the remote partner may agree that a new time frame starts with every minute, every minute past 20 seconds and every minute past 40 seconds. In this example, we assumed that time frames are 20-seconds-long. The length of time frames may depend on how well the user's watch is synchronized with the TSA. In case of precise watches, the protocol is viable with shorter (e.g. 5-seconds-long) time frames too. (See the work of Sánta for description of a similar experiment. [18]) If the user's watch cannot be synchronized with the TSA, then longer time frames (of 30 seconds, 1 minute or even 10 minutes) can be used. Thus, there is a possible tradeoff between speed and time synchronization.

If user U would like to send the signal of card C to the remote partner R using the malicious terminal T , she needs to engage in the following protocol:

1. The user inserts card C into the reader.
2. $C \rightarrow T: r$ /where r is a fresh T random number generated by C /
3. $T \rightarrow TSA: r$
4. $TSA \rightarrow T: timestamp(r)$

5. $T \rightarrow C: \text{timestamp}(r)$
6. $C \rightarrow T: \text{sign}_C(\text{timestamp}(r))$ /if the timestamp is valid/
7. $T \rightarrow TSA: \text{sign}_C(\text{timestamp}(r))$
8. The user removes the card from the reader.
9. $TSA \rightarrow T: \text{timestamp}(\text{sign}_C(\text{timestamp}(r)))$
10. $T \rightarrow R: r, \text{timestamp}(r), \text{sign}_C(\text{timestamp}(r)), \text{timestamp}(\text{sign}_C(\text{timestamp}(r)))$
11. R accepts the signal of the user if the digital signature of C is correct and both timestamps are within the same time frame (i.e. $\exists i$, where both time stamps are between t_i and t_{i+1}).

Note that most steps of the above protocol can be automated. In fact, the user needs to perform only two actions for sending a signal: she has to insert her card before she would like to send a signal, and she has to remove her card afterwards.

The user initiates the protocol by inserting card C into the card reader of the terminal. Steps 3 to 9 has to be performed in the same timeframe, otherwise the remote partner will reject the signal. After Step 9, the untrusted terminal obtains the signal. The signal itself is the message that the untrusted terminal sends to the remote partner in Step 10.

Using the above protocol the user may send various signals to the remote partner who can interpret a series of signals as a single message. There are two major principles the user and the remote partner must adhere to:

P1 User: After the user started sending a message she has to send a signal in every time frame until the end of the message. The user must not send more than one signal in a time frame, and she must not insert more than one card in a time frame into the card reader of the terminal.

Recipient: Every time frame in the message may contain one and only one signal. If it is not so, the remote partner should consider that the message has been tampered with.

P2 User: Apart from the signals used for transmitting messages, the user has to clearly mark the beginning and the end of the message. Otherwise the untrusted terminal could tamper with the message by chopping signals off from either the beginning or the end of the message. One possibility for marking the beginning and the end of the message is sending a special *startstop* signal (using a dedicated smart card). The other possibility is to use a special combination of signals that may not appear during the message.

Recipient: If the remote partner receives a message that does not have a *startstop* symbol at the beginning or it is not terminated by a *startstop* symbol, then the remote partner should consider that the message has been tampered with.

In other words, messages should be started by and terminated by *startstop* signals, and they may not contain any empty time frames. Each time frame in a message may contain one and only one signal.

The user is able to send binary messages of an arbitrary length if she has three smart cards: a card called *one* for sending message bit '1', a card called *zero* for sending message bit '0', and a *startstop* card for marking the beginning and the end of messages. See Figure 3 for an example.

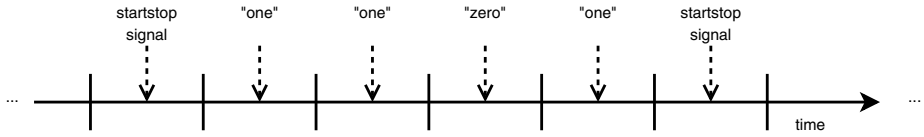


Fig. 3. An example

There is a possible tradeoff between the number of cards and the speed of protocol. The user and the remote partner may agree on a different set of signals, and on a different way of encoding the message into signals. For example, when sending numbers, the user may have one card for sending number '1', another for sending number '2', '3', etc. In case of sending text messages, it may be beneficial to select an encoding optimized for sending text.

In the next section, we are going to show that this protocol is secure against a certain attack model.

6 Analysis

We consider that the attacker controlling the terminal has possibilities described by the attack tree below. [19] The aim of the attacker is to make the remote recipient accept a message that the user did not sign as an authentic message originating from the user. (See Section 3 for the model of the attacker.)

1. The attacker may insert additional signals into a message.
2. The attacker may remove one or more signals from
 - (a) the beginning of a message.
 - (b) the end of a message.
 - (c) the middle of a message (i.e. from parts other than the beginning and the end of the message).
3. The attacker may modify the message by changing one or more signals into different ones. The attacker can do this by removing a signal from the message and inserting a different one instead by
 - (a) obtaining a signature from a card that is used for sending a different signal, and obtaining the necessary timestamps from the TSA.
 - (b) forging a signature of a card that is used for sending a different signal, and obtaining the necessary timestamps from the TSA.
 - (c) obtaining a signature from a card that is used for sending a different signal, and forging the necessary timestamps of the TSA.
 - (d) forging a signature of a card that is used for sending a different signal, and forging the necessary timestamps of the TSA.
4. The attacker may perform cut-and-paste attacks.

Proposition 1. *If our assumption in Section 3 hold, then the solution we proposed in Section 5 is secure against the attacks in the above attack tree.*

Proof. We show that – according to our assumptions – none of the attacks on the leaf nodes of the attack tree are possible.

- Attack 1 is not possible. Between the starting and the terminating *startstop* signals, every time frame in the message contains one and only one signal. If the attacker inserts an additional signal, then one of the time frames contain more than one signal (Principle P1 is violated), so the message is invalid and is rejected by the recipient.
- Attack 2a is not possible. If the attacker removes the first *startstop* signal, then the message becomes invalid (because Principle P2 is violated), so it is rejected by the recipient.
- Attack 2b is not possible. If the attacker removes the terminating *startstop* signal, then the message becomes invalid, because Principle P2 is violated. Invalid messages are rejected by the recipient.
- Attack 2c is not possible. If the attacker removes any *one* or *zero* signals from the message, then an empty time frame appears in the message, so it becomes invalid because Principle P1 is violated. Invalid messages are rejected by the recipient.
- Attack 3a is not possible. According to Principle P1, the user does not allow the terminal to communicate with two cards in the same time frame. This means that the terminal is unable to obtain signature from a different card in the same timeframe. The terminal is able to obtain signature from a different card in a different time frame. Since both timestamps in a signal must reside in one time frame, the attacker cannot delay any step in the protocol for sending signals to make the valid signal appear in a different time frame.
- Attack 3b is not possible, because the attacker is unable to forge the signature of an unknown private key (Assumption 5) and the attacker cannot extract keys from smart cards (Assumption 6).
- Attack 3c is not possible, because the attacker is unable to forge the signature of an unknown private key (Assumption 5), and the attacker cannot extract the key from the TSA.
- Attack 3d is not possible, because neither Attack 3b nor Attack 3c is possible.
- Attack 4 is not possible, because each signal contains a timestamp that clearly marks time frame of the signal.

7 Conclusion

We propose that – using multiple smart cards – the user can sign or authenticate messages even at malicious terminals. The user can achieve this by blocking the communication channel between the terminal and certain cards and by allowing the terminal to communicate with other cards. It is not the datablocks she signs that contain the content of the message (because she has no means to guarantee the integrity of these datablocks before they are signed), but she encodes this content as a set of cards she uses for signing. In fact, the user signs her message by inserting smart cards into the reader of the terminal and by removing them.

We showed a protocol where she has a watch and she can access a trusted time stamping authority (TSA). Using this protocol, she can send messages of any length with a constant amount of cards. We have also proven that the protocol is secure against a certain attack model.

The solution we propose might sound awkward. However, we do not know of any other solution that allows the average human user to send long authentic messages without a trusted terminal.

References

1. Abadi, M., Burrows, M., Kaufman, C., Lampson, B.: Authentication and Delegation with Smart-cards. Theoretical Aspects of Computer Software: Proc. of the International Conference TACS'91, Springer, Berlin, Heidelberg (1992)
2. Schneier, B., Shostack, A.: Breaking up is Hard to do: Modelling security threats for smart cards. USENIX Workshop on Smart Card Technology, Chicago, Illinois, USA, <http://www.counterpane.com/smart-card-threats.html> (1999)
3. Balfanz, D., Felten, E.: Hand-Held Computers Can Be Better Smart Cards. Proceedings of USENIX Security '99 Washington, DC. (1999)
4. Gobioff, H., Smith, S., Tygar, J.D.: Smart Cards in Hostile Environments. In Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Nov 1996, 23-28 (1996)
5. Clarke, D., Gassend, B., Kotwal, T., Burnside, M., Dijk, M.v., Devadas, S., Rivest, R.: The Untrusted Computer Problem and Camera-Based Authentication (2002)
6. Naor, M., Pinkas, B.: Visual Authentication and Identification. Lecture Notes in Computer Science, vol 1294 (1997)
7. Matsumoto, T.: Human-Computer cryptography: An attempt. In ACM Conference on Computer and Communications Security, pp 68-75 (1996)
8. Schneier, B.: The Solitaire Encryption Algorithm. <http://www.counterpane.com/solitaire.htm> (1999)
9. Stabell-Kulo, T., Arild, R., Myrvang, P.: Providing Authentication to Messages Signed with a Smart Card in Hostile Environments. Usenix Workshop on Smart Card Technology, Chicago, Illinois, USA, May 10-11, 1999. (1999)
10. Asokan, N., Debar, H., Steiner, M., Waidner, M.: Authenticating Public Terminals. Computer Networks, 1999 (1999)
11. Berta, I.Z., Vajda, I.: Limitations of humans at malicious terminals. Tatra Mountains Mathematical Publications, vol. 29, pp 1-16 (2004)
12. Berta, I.Z., Butty n, L., Vajda, I.: Privacy protecting protocols for revokable signatures. Cardis2004, Toulouse, France (2004)
13. Berta, I.Z., Butty n, L., Vajda, I.: A framework for the revocation of unintended digital signatures initiated by malicious terminals. IEEE Transactions on Secure and Dependable Computing, vol. (Vol. 2, No. 3), pp. 268-272, July-September (2005)
14. Gruschka, N., Reuter, F., Luttenberger, N.: Checking and Signing XML Using Java Smart Cards. CARDIS 2004, Toulouse, France (2004)
15. Girard, P., Giraud, J., Gauteron, L.: Secure electronic signatures when Trojan Horses are lurking. e-smart 2004, Sophia Antipolis (2004)
16. Berta, I.Z., Vajda, I.: Documents from Malicious Terminals. SPIE Microtechnologies for the New Millenium 2003, Bioengineered and Bioinspired Systems, Spain (2003)
17. Maurer, U.: A Unified and Generalized Treatment of Authentication Theory. Proc. 13th Symp. on Theoretical Aspects of Computer Science (STACS'96), Lecture Notes in Computer Science, Berlin: Springer Verlag, vol 1046, pp. 387-398, 1996. (1996)
18. S nta, r.: Smart Card based Application for Message Authentication in a Malicious Terminal Environment. Master's Thesis, Fachhochschule Darmstadt (2004)
19. Schneier, B.: Attack Trees. Dr. Dobb's Journal December 1999, <http://www.schneier.com/paper-attacktrees-ddj-ft.html> (1999)

Diverging Keys in Wireless Sensor Networks

Michał Ren^{1,*}, Tanmoy Kanti Das², and Jianying²

¹ Adam Mickiewicz University
Poznań, Poland

`renmich@amu.edu.pl`

² Systems & Security Department
Institute for Infocomm Research

21 Heng Mui Keng Terrace, Singapore 119613
{tkdas, jyzhou}@i2r.a-star.edu.sg

Abstract. Currently, the most popular ways of dealing with the key distribution problem in sensor networks are random predistribution schemes. For relaxed, realistic assumptions about the attacker, the key infection protocol [1] is also available. In this paper, by accepting the relaxed assumptions from [1], we propose a scheme which makes pairwise keys “drift” or diverge, which enhances security and can be used as a key distribution method. The most notable feature of this scheme is that, under some assumptions about the sensor nodes, it incurs no communication overhead at all.

Keywords: Sensor Network Security, Node Compromise Attack, Diverging Keys.

1 Introduction

Sensor networks, that is, wireless networks of small nodes with sensing abilities are becoming more and more prevalent in a wide variety of applications. These networks are deployed mostly in inaccessible and hazardous conditions. They are also widely used in military applications in a hostile environment. As technology matures, there is a tendency for the nodes to become cheaper and more capable. Thus they will become attractive for home use also.

The security of these networks is a prime concern for the application developers if they are deployed in military applications or in critical applications like medical monitoring. In addition to node compromise and jamming of communication channel, several other attacks can be mounted on the sensor network, notably Sybil Attack [10], Wormhole attack [7], Node Replication Attack [11], Denial of Message Attack [9], etc.. In the Sybil attack, a single node takes on multiple identity to deceive other nodes. In the wormhole attack, presented in [7], the attacker captures message bits at one location and replays them in another location. They also present an algorithm, known as packet leashes, to defeat

* The first author’s work was done while he was visiting Institute for Infocomm Research (I^2R) in 2005 under I^2R ’s sponsorship.

such attacks. In one recent paper, McCune et al. [9] highlighted the Denial-of-Message (DoM) attack and proposed detection algorithm to thwart the attack. In the DoM attack, a set of nodes act maliciously and prevent broadcast messages from reaching certain section(s) of the sensor network. Authors [9] propose Secure Implicit Sampling algorithm to detect such attacks. In the node replication attack, cryptographic secrets from the compromised sensor nodes are used to create duplicate sensor nodes in large number. Then these sensor nodes are placed in critical locations of the sensor network to mount the attack. Several protocols were proposed to defend the sensor network from the replication attack and most promising among them is *distributed detection protocol* [11]. These attacks present some interesting security challenges, especially for key management and distribution.

A typical sensor node is neither tamper proof, nor suitable for public-key cryptography as they possess very little computing capability. Thus the symmetric key cryptography is the preferred solution for the application developers where each node shares a key with another neighbour for secure communication. In most of the applications, it is impossible to predict how the nodes will be positioned with respect to one another, thus predistribution of keys shared between the neighbours is not possible. The other option of loading each node with all possible shared keys is impossible because of memory constraints. However, random key predistribution is possible.

Most of the papers, available in the existing literature, assume a strong adversary model. In this model, it is assumed that the attacker is present prior, during, and after the deployment of the network and the attacker possesses enough resources to monitor all the communications of the network at all times. Furthermore, it is assumed that the attacker may compromise a few nodes and turn them into malicious ones. Such strong assumptions are suitable for military and other mission-critical applications. However, for commodity sensor networks such assumptions may not be practical. Also the use of resource hungry protocols in the strong adversary model increases the cost of deployment which may discourage everyday use of sensor networks. Relaxing the assumptions about the attacker to be more realistic, namely, assuming that the adversary cannot eavesdrop on all communication links all the time, allows for development of counterintuitively secure protocols for commodity sensor networks [1].

In this paper, by accepting the relaxed assumptions from [1], we propose a scheme which makes pairwise keys “drift” or diverge, which enhances security and can be used as a key distribution method. The most notable feature of this scheme is that, under some assumptions about sensor nodes, it incurs no communication overhead at all.

The rest of the paper is organized as follows. We review the related work in Section 2. After that, we present the key divergence protocol in Section 3, and discuss the security and benefits of our protocol in Section 4. We conclude in Section 5.

2 Related Work

In the past, many solutions have been proposed to the problem of key distribution and management in sensor networks. Most of the recent work has been focused on random predistribution schemes [5,8], first proposed by Eschenauer and Gligor [6]. Let us recall the phases of the basic scheme:

1. Key predistribution phase is conducted offline. It consists of generating a large pool of keys and loading a small amount of different randomly drawn keys into each node. Every key should also have an assigned identifier.
2. Shared key discovery phase takes place in the target environment, after the sensor nodes are deployed. Every node discovers its neighbors, and tries to establish a common key with every one of them. The simplest method of achieving that is for every node to broadcast, in plaintext, the list of identifiers of keys that it possesses. This phase establishes network topology, as two nodes are “linked” only if they share a common key.
3. Path-key establishment phase allows the nodes that are neighbors (that is, are within wireless communication range of each other) but do not share a common key from the pregenerated pool, to establish a common path-key.

Another solution to the key distribution and management problem is proposed by Anderson, Chan and Perrig in [1] — the key infection protocol relies on weakening of the assumptions about the attacker. In most of the research papers, it is assumed that the attacker is very powerful and capable. Most of these assumptions are directly adopted from cryptography and based on the experience of World War II. In those days, communication resources were limited, and it was possible to monitor all the communications all the time. Thus came the assumption of strong adversary. Now consider that a large number of sensor nodes are dropped from air into hostile territory. If there is no prior knowledge of deployment, it is practically impossible for the enemy to be readily present there with their own sensor network. Another obstacle that deters the enemy from using this type of targeted surveillance is the limited battery-life of these devices. However, this type of targeted surveillance is possible for high value targets. Commodity sensor network for non-critical applications have completely different threat perception and it is simply not economical to attack them using universal surveillance. Thus the attack model considered in [1] is as follows:

1. The deployment site is inaccessible to the attacker during deployment of the network.
2. During deployment, the attacker can only monitor a small fraction of all communications. In real-world scenarios, in a couple of seconds immediately after the deployment the attacker will not be able to eavesdrop on all the communications, but only intercept a fraction of them.
3. Similarly, the attacker cannot mount active attacks during the deployment.

Thus it is assumed that the attacker is fully capable except during the deployment of the network. This enables a counterintuitive protocol in which the nodes agree on their pairwise keys by broadcasting them in the clear.

In the paper of Anderson et al. [1], authors assume that the adversary may not be able to listen all communications all the time. This may be true. However, attacker may be able to listen to a fraction of all communications and this will compromise some of the shared keys transmitted in the clear, rendering those links vulnerable. Thus pre-loading all the nodes with a master key will prevent the attacker from listening to initial communication as it will be encrypted using that key. After two nodes setup a shared key, the master key will no longer be used for encrypting any communication. If the shared keys diverge with time and communication, the attacker will find little use of compromised keys as they would need to listen to all the communications to re-compute the new shared keys generated from the compromised keys. As the attacker is unable to listen to all the communications all the time, this type of key divergence will improve the security. Also, the same key divergence protocol can also be used for generating shared secrets from the “master key”.

In this paper, we present a protocol in which the keys that the nodes possess change continuously, therefore forcing the adversary to keep monitoring all the communication links all the time. If the adversary is unable to monitor all the traffic immediately after deployment, this key divergence protocol can also be used for key distribution. As it incurs no communication overhead, it can be used with every other protocol for very little additional energy cost, provided that the nodes are able to perform symmetric encryption/decryption efficiently. We make a similar assumption regarding the adversary as done in [1], that the adversary is unable to monitor all communication links all the time. For key distribution, it is sufficient to assume that the adversary is unable to monitor all the communications right after the network deployment.

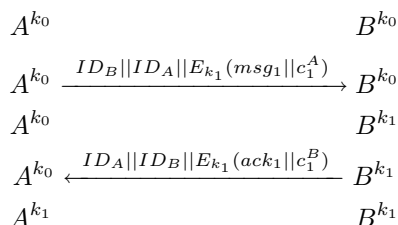
3 Diverging Keys

Computation and cryptography on sensor node is made difficult by their lack of tamper resistance and the limited amounts of energy available. The latter is especially limited, and every joule must be carefully metered. Most of the protocols devised for sensor networks are optimized for the limited resources available [2,3,4,12]. For a typical sensor node, the most expensive operation in terms of energy expenditure is wireless communication. For a small node, sending a bit can be many orders of magnitude more expensive than encrypting it by symmetric encryption. (Public-key operations are currently regarded as too expensive to be used on sensor nodes.) For instance, Carman, Kruus and Matt reported in [2] that the energy cost of just $9nJ$ per bit for AES encryption, and $21\mu J$ for sending a bit; a difference of over three orders of magnitude. It can only be expected that the gap will increase as processor technology progresses. In fact, on modern, specialized hardware, the energy cost of encrypting one bit is just $60pJ$ [13]. It is therefore imperative to minimize communication, rather than the amount of symmetric encryption/decryption. In this scenario, we propose a protocol that enables key distribution and increases link security without any communication overhead.

3.1 Basic Scheme

1. The sensor nodes are deployed with a common master key. Strictly speaking, the master key is not necessary, as it is sufficient that the nodes are able to agree on pairwise shared keys. We discuss this in detail later.
2. Each sensor node establishes communication with its neighbors, and allocates enough space to hold one key, and two counters (to prevent replay attacks) for every neighbor — one for sent and one for received messages. At the beginning, every such pairwise key is identical and is a copy of the common master key from step 1.
3. When communicating with any of the neighbors, each sensor node encrypts the message, but with the key that differs by one bit from the pairwise key that it shares with the neighbor (unless the preceding exchange has failed, in which case the key should be used as-is). The position of the flipped bit should be chosen at random. Then, the node transmits the encrypted message, preceded by the identifier of the node that it wants to communicate with (to wake it up), and by its own identifier (so the receiving node knows which key to use). It is assumed that the message will contain a counter to prevent replay attacks. The node then waits for a response in step 6.
4. Upon receiving its identifier, the node listens to the following message, and then tries to discover the new key by brute-force attack, starting with the previous key. Of course, since the new key can differ only in one bit from the previous key, the maximum number of tries is equal to the length of the key in bits, and on average, only half as many tries will be needed. If the previous exchange has failed and was initiated by this node, the node may also try every key that differs by one bit from the key that it wanted to establish in the failed exchange.
5. If the listening node succeeds in decrypting the message, it checks if the counter is greater than the one held in memory. If it is not, the message is discarded as an attempted replay attack. If the message is fresh, the node updates the counter, changes the pairwise key that it shares with the sending node, and replies using the new key. The old key is discarded.
6. After receiving the response, encrypted with the new key, the first node also discards the old key and replaces it with the new one. If an incorrect response is received, or none at all, the node considers the key change to have failed.
7. Steps 3–6 are repeated when the nodes wish to communicate again, for as long as the network is active.

The following diagram illustrates a typical exchange, showing the keys that are in possession of the nodes during each step:



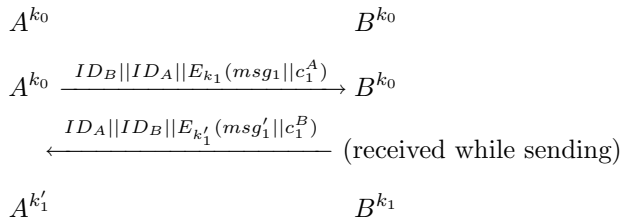
Legend:	
	: concatenation
ID_X	: identifier of node X
msg_i	: message i
ack_i	: acknowledgment to msg i
$E_k(m)$: m encrypted with key k
c_i^X	: counter sent by node X

We have described the protocol assuming the presence of a master key (step 1, 2). However, the scheme can operate on top of any random key predistribution scheme. In that case, the pairwise shared key used for communication start to diverge. Thus the steps 1 and 2 are replaced by the following step.

1. Each node discovers the shared keys that it share with other node using any random key predistribution scheme. This phase establishes network topology, as two nodes are “linked” only if they share a common key.

Several assumptions must be made about the message structure in order for this scheme to work. First of all, it is assumed that the messages are subjected to error detection code or other means of checking their integrity; otherwise, there will be no easy way to check if the brute-force in step 4 has succeeded or not. There should be enough redundancy to make the probability of a false-positive (when the node has the wrong key, but it thinks it decrypts the message correctly) sufficiently small. The messages should also contain an incrementing counter, to prevent a replay attack. Note that, our scheme is not designed to be forward secure as the new key is dependent on the old key.

Another assumption that must be made but which holds for most sensor networks, is that no node may execute different steps of the protocol at once, especially no simultaneous sending and receiving of messages is possible. Otherwise, communication may be permanently severed. Consider the following diagram:



Both A and B will start receiving each other’s message at the same time and they might start executing steps 4 and 5, without completing step 3. Then, node A will change the key from k_0 to k_1' , discarding k_0 , and B will change the key from k_0 to k_1 , also discarding k_0 . So A and B will have two different keys, and will never be able to communicate again. However, if a sending node completes step 3, it then expects a reply as per step 6, encrypted with the new key; otherwise the key exchange will fail. Obviously this scenario is theoretical in nature at present, as current generation sensor nodes can not send and receive

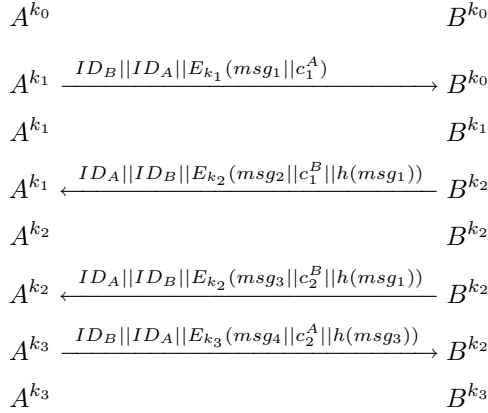
simultaneously. However, in the future, it may become possible for sensor nodes to send and receive at the same time. We will present a variant of the present scheme which can effectively deal with such situations without assuming that the operations must be sequential.

3.2 A Variant

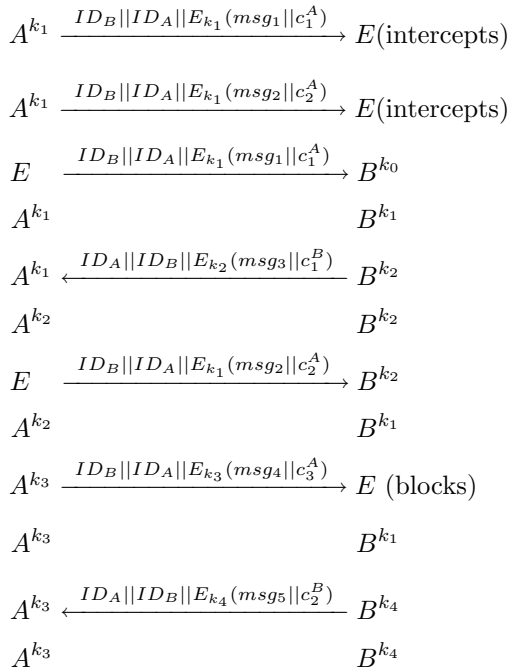
A major goal of our protocol is to minimize the communication overhead in key distribution. However, the acknowledgment required in the basic scheme is indeed an overhead for many networks where acknowledgment for communication is not used. We propose a variant of the protocol, in which the requirement of the acknowledgment message in every step is replaced with much more modest addition of a counter value of previous message. We will make use of a technique similar to one used when an “ack” message gets lost. The scheme is as follows:

1. The sensor nodes are deployed with a common master key.
2. Each sensor node establishes communication with its neighbors using a pairwise key which is a copy of the master key. Each node also keeps a pair of counters for every neighbor and a pair of hashes of the last message sent to and received from every neighbor. Note that, one can also use the counter value in place of hash value.
3. When communicating with any of the neighbors, each sensor node encrypts the message using a key that differs by one bit from the pairwise shared key. However, **present node can not change the key if the previous key change is initiated by it**, in that case, same key is used for encryption. Then, the node transmits the encrypted message, preceded by the identifier of the node that it wants to communicate with (to wake it up), and by its own identifier (so the receiving node knows which key to use). It is assumed that the message will contain a counter to prevent replay attacks, and a counter/hash of the previous message received from that node. (We will discuss the necessity of the hash below.) Old key is kept in the memory to recover from potential loss of message.
4. Upon receiving its identifier, the node listens to the following message and then tries to discover the new key by brute-force attack starting with the previous key. Of course, since it knows the previous key, the maximum number of tries is equal to the length of the key in bits, and on average, only half as many tries will be needed. The node may also try every key that differs by one bit from the “old key”.
5. If the listening node succeeds in decrypting the message, it checks if the counter is greater than the one held in memory. If it is not, the message is rejected as a replay attack. If the message is fresh, the node updates the counter, and checks if the hash corresponds to the last message sent to that node (unless that is the first message received from that node). If it does, the node changes the pairwise key that it shares with the sending node. The previous pairwise key is now remembered as the “old key”.
6. Steps 3-5 are repeated when the nodes wish to communicate again, for as long as the network is active.

This scheme does not require any acknowledgment message. However, here communicating nodes can only change the shared key alternatively. If any one node sends more than one message without receiving a message from the other node, all the messages will be encrypted using the same key. Thus while communicating, a pair of nodes change the key alternatively. Let us illustrate it with the following diagram.

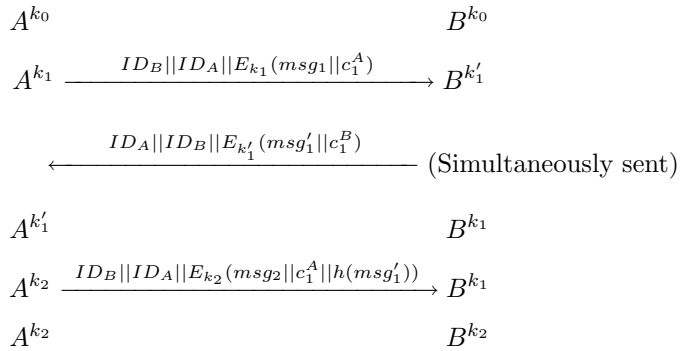


Note that, when node B sends two consecutive messages there is no change in the key. Furthermore, nodes may change the key only if received hash matches the one from last sent message. Consider what might happen if message hashes were not checked:



Here, eavesdropper E records two messages from the node A, then replays the first one, waits until node B responds trying to change the key and replays the second intercepted message. To node B, it looks like its previous message got lost in transit, and so it changes the key as per node A’s former request. Now, both nodes think that they should initiate the next key change. E waits until both A and B do that, blocking messages if necessary. Once both nodes do so, they are not able to communicate again. On the diagram, in the last phase, node A has key k_3 , and remembers k_2 as the “old key”, while B has key k_4 and remembers k_1 . Thus, communication would be lost.

Now, let us discuss how this scheme can handle simultaneous message communications.



After the simultaneous message exchange, we have a key disagreement. Let us analyze what happens at node A. Keys are changed in following sequence: $k_0 \rightarrow k_1 \rightarrow k'_1$. In this protocol, we always store the old key, so two stored keys are k_1 (old key) and k'_1 (present key). Similarly at node B, two stored keys are k'_1 (old key) and k_1 (present key). Note that at this point both the node A and B think that they can change the key as the last key change is not initiated by either. However, we regain the key synchronization with the next message. Let’s assume that node A wishes to send the next message and encrypts it with key k_2 which differs from k'_1 (present key at node A) by one bit. After receiving the message, node B will first try to decrypt it using all the keys that differ by one bit from k_1 . This will fail, and the node will then try using all the key that differ by one bit from the old key k'_1 . This time it will be able to decrypt the message successfully and will change the key to k_2 . Thus, key synchronization will be regained.

4 Further Discussions

4.1 Benefits

In any random key predistribution scheme, nodes are preloaded with a small number of keys selected from a large pool of keys. It is presumed that any two nodes willing to communicate will have at least one common key. The problem with this approach is that a single node capture affects many other nodes due

to the presence of common keys between the captured node and un-captured ones. This is a real problem with any key predistribution scheme. However, our scheme is resilient to such node compromise. Only the keys used for communicating with the captured nodes are affected. All other nodes can continue to communicate without any hindrance. Another advantage of our scheme is that the adversary needs to continuously record messages from the compromised nodes; otherwise the keys will change, making their re-discovery by the attacker difficult or impossible, unless the attacker has physical access to the memory of compromised nodes. Some predistribution schemes, where each node stores a pairwise key for every node present in the network, are not scalable. In contrast, our scheme is scalable. One can easily add our scheme over any key predistribution scheme.

The main objective of the proposed scheme is making the pairwise keys in the nodes diverge over time, thus safeguarding against node capture. It might seem that the same effect could be realized by a simpler protocol, such as transmitting a new random key, encrypted with the old one, at predetermined intervals. However, the proposed protocol offers significant advantages over such a simplistic scheme, the foremost being the lack of communication overhead, or — in case of the variant protocol — only modest overhead caused by transmitting message hashes or additional counters.

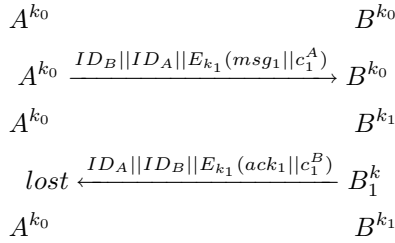
The energy cost of adding the basic scheme on top of normal communications between nodes is quite low, provided that the messages exchanged between sensor nodes already use error detection codes and counters. If that is the case, there is no extra communication overhead at all (or in the case of the variant scheme, only slight additional energy cost of transmitting short message hashes or additional counters), and the only additional energy cost is the brute-force breaking of the key. If we assume $60pJ/bit$ for encryption/decryption [13], $21\mu J/bit$ for sending [2], 128-bit keys and blocks, then the cost of one bit "flip" (128 bits, and 64 decryptions on average) will be only about $0.5\mu J$. We can compare it to the cost of generating a new key, encrypting it with the old one and just transmitting it. However, the transmission of just 128 bits (key only, no node IDs, error correction, etc.) will cost about $2500\mu J$ — which would be enough for about five thousand bit "flips". The difference is likely to increase even more in the future.

The memory cost is also quite low — just the cost of storing the pairwise keys for all neighbors, two counters for every neighbor, some space for temporarily storing old keys, and possibly (in the variant scheme) a hash for every neighbor.

4.2 Security

One vulnerability of the scheme is that if the neighboring nodes ever disagree on their pairwise key by two bits or more, they may never communicate again. However, there is no way for the attacker to cause that, short of breaking the key, and in case of normal communication failures everything shortly returns to normal as explained. It might seem that in the basic scheme, loss of the

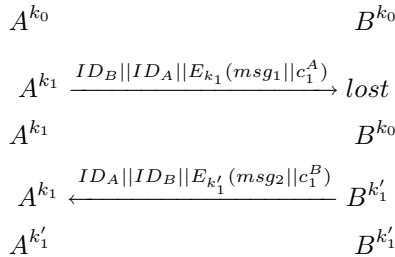
acknowledgment message could cause that. However, the scheme can handle that as follows:



Legend:
: concatenation
ID_X : identifier of node X
msg_i : message i
ack_i : acknowledgment to msg i
$E_k(m)$: m encrypted with key k
c_i^X : counter sent by node X

After the acknowledgment message is lost, node A thinks the pairwise key is k_0 and node B thinks the key is k_1 . Everything returns to normal with the next message. If node A sends it, it will use key k_0 , because the last exchange with B has failed (see step 3). Since k_0 differs from k_1 by one bit, B will decrypt successfully. If node B sends the next message, node A will also decrypt it successfully, since it will try all keys differing by one bit from k_0 and k_1 (see step 4).

Let us now consider the situation in the variant scheme where we require no acknowledgment for key divergence:



Thus when the first message sent by node A get lost, node B thinks that the shared key is k_0 , but the key is k_1 at node A, so there is mismatch of keys between node A and B. However, when node B randomly changes key k_0 to k_1' and sends an encrypted message to A, node A is able to decrypt the message because node A does not only try all keys that differ by one bit from k_1 but also all that differ by one bit from the “old key” k_0 (step 4). Thus, node A is able to regain key synchronization. Also, if A decides to send the message after the lost message, it cannot change the key and so encrypts the message using the same key k_1 . B is able to decrypt the message as k_1 differs from k_0 by one bit only. B then changes the key to k_1 , regaining synchronization.

Under common omnipresent adversary assumption, this scheme pays only a relatively small penalty in energy consumption and memory capacity, accomplishing essentially nothing. However, under a more realistic set of assumptions [1], where the attacker is unable to observe all communications, it succeeds in performing safe key distribution. It also offers the following fringe benefits:

- Even if the adversary monitors a part of the network, he can never cease to do so, or the nodes will change keys while he is not looking.
- The keys on high-traffic links naturally change faster, so the security in the critical parts of the network is higher.

Let us now consider a case of adding the present scheme over a random key predistribution scheme. In random key predistribution scheme, every node is pre-loaded with keys selected randomly from a large pool of keys. After network deployment, nodes trying to communicate first identify a common key that they both possess to encrypt the messages. However, these schemes, as stated earlier, are vulnerable to node capture attacks. Because same keys are present in many nodes, and capture of single node exposes several other links where keys present in the captured node were used. Now, if we add the key divergence protocol on top of random key predistribution, then each pair-wise key used for communication will start diverging with the communication. So, if a node is compromised, no key already in use elsewhere will be similar to the keys stored in the memory of captured node. Thus, except for the links passing through the captured nodes, no other link will be exposed. Therefore, our scheme is able to rectify a significant vulnerability of existing random key predistribution schemes.

5 Conclusion

In this paper, we have proposed a key divergence protocol having no (or very little) communication overhead. In some cases, even the modest energy cost of this protocol could be too much for a particular sensor network. In that case, present protocol can be suitably modified so that every message only has a certain chance of introducing a new key, thus reducing the overhead further. Also the nodes can always start brute force checking with the last key, so that if it was not changed, only one decryption is necessary. In addition, this scheme can be used with other means of key distribution, essentially acting as link security enhancement. For instance, it can be used with the key infection protocol [1], or, as already explained, one of the random key predistribution protocols [5,6,8]. Furthermore, proposed scheme when used with any random key predistribution scheme, is able to thwart the node capture attack to the extent that communication links, other than those passing through the compromised nodes, remain unaffected.

References

1. R. Anderson, H. Chan, and A. Perrig. Key infection: Smart Trust for Smart Dust. In *Proceedings of IEEE International Conference on Network Protocols (ICNP 2004)*, October 2004.

2. D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and Approaches for Distributed Sensor Network Security. Technical Report 00-010, NAI Labs, Cryptographic Technologies Group Trusted Information Systems, NAI Labs, The Security Research Division Network Associates, Inc. 3060 Washington Road (Rt. 97) Glenwood, MD 21738-9745, 2000.
3. H. Chan and A. Perrig. PIKE: Peer Intermediaries for Key Establishment in Sensor Networks. In *The 24th Conference of the IEEE Communications Society (Infocom 2005)*, March 2005.
4. H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 197-213, Washington DC, USA, 2003. IEEE Computer Society.
5. W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A Pairwise Key Predistribution Scheme for Wireless Sensor Networks. *ACM Transactions on Information and System Security*, 8(2):228-258, 2005.
6. L. Eschenauer and V. D. Gligor. A Key-management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 02)*, pages 41-47, New York, USA, 2002. ACM Press.
7. Y. C. Hu, A. Perrig and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. In the proceedings of *INFOCOMM 2003*.
8. D. Huang, M. Mehta, D. Medhi, and L. Harn. Locationaware Key Management Scheme for Wireless Sensor Networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 04)*, pages 29-42, New York, NY, USA, 2004, ACM Press.
9. J. M. McCune, E. Shi, A. Perrig and M. K. Reiter. Detection of Denial-of-Message Attacks on Sensor Network Broadcasts. *IEEE Conference on Security and Privacy*, 2005.
10. J. Newsome, E. Shi, D. Song and A. Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. In *IPSN 2004*, pages 259-268, April 26-27, 2004, Barkley, California, USA.
11. B. Parno, A. Perrig and V. Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. In *IEEE S&P*, 2005.
12. A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Mobile Computing and Networking*, pages 189-199, 2001.
13. K. Tiri, D. Hwang, A. Hodjat, B. C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. AES-Based Cryptographic and Biometric Security Coprocessor IC in 0.18-um CMOS Resistant to Side-Channel Power Analysis Attacks. In *Proceedings of the Symposia on VLSI Technology and Circuits*, pages 216-219, June 2005.

A Generic Transformation from Symmetric to Asymmetric Broadcast Encryption*

Ulrich Huber and Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security
Ruhr-Universität Bochum, Germany
{huber, sadeghi}@crypto.rub.de

Abstract. Broadcast Encryption (BE) schemes allow a sender to efficiently encrypt messages for a large set of receivers. The currently most efficient BE schemes in the stateless receiver scenario are based on symmetric cryptography. However, a variety of business models with mutually mistrusting senders necessitates the use of asymmetric cryptography. We propose a generic framework that allows to transform a large class of symmetric BE schemes into asymmetric schemes, where the transformation employs an arbitrary hierarchical identity based encryption scheme. Applying our framework, we transform a recent symmetric scheme, called layered punctured interval scheme, for which no asymmetric version has yet been published. In addition, we give a formal proof of the chosen ciphertext security of our framework, which allows to generically transform any future symmetric BE scheme within the large class into a chosen-ciphertext-secure asymmetric scheme with the same efficiency measures.

1 Introduction

Broadcast Encryption (BE) schemes allow a sender to efficiently encrypt messages for a large set of receivers and subsequently send the ciphertext over an insecure broadcast channel. BE has found many applications such as pay-TV, distribution of copyrighted multimedia content on CD or DVD, and Internet audio/video streaming [2,3,4]. Many variants of BE schemes have been studied: The set of receivers may be static or dynamically changing; the number of revoked (or excluded) receivers may be bounded or unbounded; the receivers may have static keys, termed stateless receivers, or periodically update them, called stateful receivers; traitor tracing may be possible, meaning that cheating users can be detected when illegal receiving devices are found. We analyze the probably most general and difficult variant: the set of receivers is dynamically changing, the number of revokable receivers is unbounded, the receivers are stateless, and traitor tracing is possible. As argued by Naor, Naor and Lotspiech [5], this scenario is quite realistic and relevant for many applications.

The currently most efficient BE schemes in this scenario are symmetric key BE schemes [5,6,7,8]. The sender uses the same keys for encryption as the receivers

* This paper is an extended abstract of a technical report [1].

use for decryption. This is suitable for many applications in which there is a single trusted sender, who is in possession of all keys and thus a single point of failure for the complete scheme. However, there are practically relevant business models which cannot be implemented with symmetric BE schemes. Consider the case of several mutually mistrusting senders, such as several pay-TV providers using the same set-top box infrastructure, or several content providers broadcasting to the same CD or DVD players. In this case, it remains unclear which trusted party should be in possession of the symmetric encryption keys.

Dodis and Fazio [9] therefore transformed three symmetric BE schemes into asymmetric schemes on an individual case basis. The asymmetric version of the schemes allows the public encryption keys to be shared by any number of mutually mistrusting senders. As encryption and decryption keys are different, this does not compromise security. However, the authors do not formalize a generic approach, but individually transform three specific schemes: the Complete Subtree (CS) and the Subset Difference (SD) scheme from [5] and the Layered Subset Difference (LSD) scheme from [6]. Yet there are more recent symmetric BE schemes [7,8] for which, to the authors' knowledge, no asymmetric version exists.

1.1 Our Contribution

We propose a generic framework that allows to transform a large class of symmetric BE schemes into asymmetric BE schemes, where only the decryption keys are secret. The only required property of the symmetric scheme is that each of its secret keys must be derived from exactly one key among a finite set of master keys; we call these schemes *derivation BE* schemes. We stress that all of the most efficient symmetric BE schemes are derivation BE schemes [5,6,7,8]. As proof of concept, we instantiate our framework with two BE schemes; namely, the SD scheme of [5] and the Layered Punctured Interval (LPI) scheme of [7]. Although the authors of [9] already transformed SD into an asymmetric BE scheme, we show, to the best of our knowledge, the first asymmetric version of LPI. Our framework employs an arbitrary Hierarchical Identity Based Encryption (HIBE) scheme; if instantiated with the currently best HIBE scheme [10], the efficiency measures of the symmetric BE scheme, such as transmission overhead and receivers' key storage size, directly transfer to its asymmetric version.

In addition, we give a formal proof of the IND-CCA1 security of our framework. Although Dodis and Fazio provided reasonable arguments for the existence of such a proof, they did not formalize their arguments into an actual proof [9]. The combination of our framework and security proof allows to transform any future symmetric BE scheme within the large class to be generically transformed into an IND-CCA1-secure asymmetric scheme.

The main idea is to replace the pseudo-random key trees used by the derivation BE scheme with a hierarchical private key tree of an HIBE scheme. The efficient storage size of any derivation BE scheme, which mainly depends on key derivation in pseudo-random chains, thus transfers to its asymmetric version. The constant ciphertext size of the HIBE scheme [10] ensures an identical transmission overhead of the asymmetric version. The major task in the definition of

our framework is to find an efficient mapping that maps the pseudo-random key trees in the original scheme to a hierarchy of identifiers in the HIBE scheme.

2 Related Work

Fiat and Naor were the first to formally define the functional requirements of a BE scheme [11]. Naor et al. introduced a generic class of BE schemes called *Subset Cover BE* (SCBE) schemes, which cover the set of non-revoked receivers with a collection of well-defined subsets [5]. Specifically, they proposed two SCBE schemes based on binary key trees: CS and SD. With the LSD scheme, Halevy and Shamir reduced the storage size of receivers at the price of doubling the header length [6]. Two recent SCBE scheme based on one-way chains are due to Jho et al. [7,8]: They introduced the layered punctured interval scheme in [7] and the tree-based circle scheme in [8]. However, to the best of our knowledge no asymmetric version of these schemes has yet been published.

A trivial transformation from symmetric to asymmetric BE schemes results in huge private keys due to the loss of key derivation capabilities [5,9]. Dodis and Fazio therefore non-trivially extended CS, SD, and LSD to the asymmetric setting using (hierarchical) identity based encryption schemes [9]. Other existing asymmetric BE schemes can only revoke a bounded number of receivers, which needs to be fixed a-priori in the setup phase [12,13,14]. This is contrary to our focus on the unbounded scenario discussed above. Moreover, the transmission overhead of these schemes grows with the a-priori bound, whereas that of the symmetric schemes [5,6,7,8] only grows with the number of actually revoked receivers.

Boneh et al. propose an asymmetric BE scheme for unbounded revocations, where ciphertext and private keys have constant size [15]. However, during decryption even the receivers need the public encryption key, which grows linearly with the number of receivers, whereas ours has constant size and only applies for encryption. In addition, their attack model is non-adaptive, whereas ours is adaptive. Finally, their scheme does not provide traitor tracing capabilities, whereas our framework maintains those of the original schemes, e.g., [5,6].

3 Preliminaries

3.1 Roles in Our System Model

In symmetric BE schemes, there is a single *sender* that often coincides with the trusted (*broadcast*) *center*, which manages the broadcast channel and distributes key material. The N *users* obtain content via devices that we refer to as *receivers*; the set of receivers is $\mathcal{U} := \{u_i \mid 1 \leq i \leq N\}$. When a user violates the terms and conditions of the application, the center revokes the keys in this user's receiver and thus makes them useless for decryption purposes. We denote the set of revoked receivers with $\mathcal{R} := \{r_1, r_2, \dots\} \subset \mathcal{U}$.

In asymmetric BE schemes, the roles of sender and center are separate. There are at least two *mutually mistrusting senders*. In addition to distribution of

the private decryption keys to receivers, the role of the center, which is still trusted by all parties, comprises distribution of public encryption keys to the senders.

3.2 Notation

We use some standard notations throughout the paper. First, we denote scalar objects with lower-case variables, e.g., o_1 , and object tuples with upper-case variables, e.g., T_1 . When we summarize objects or roles in set notation, we use an upper-case calligraphic variable, e.g., $\mathcal{O} := \{o_1, o_2, \dots\}$ or $\mathcal{T} := \{T_1, T_2, \dots\}$, where $\mathcal{P}(\mathcal{O})$ denotes the powerset of \mathcal{O} , which is the set of all subsets of \mathcal{O} . Second, let \mathcal{J} be a finite index set. Then $\{o_j | j \in \mathcal{J}\}$ denotes a set of objects indexed by \mathcal{J} . Third, let A be an algorithm. By $y \leftarrow A(x)$ we denote that y was obtained by running A on input x . Fourth, $o_1 \xleftarrow{R} \mathcal{O}$ denotes the selection of a random element of the set \mathcal{O} with uniform distribution. Last, we use upper-case calligraphic variable for roles in security proofs, e.g., \mathcal{A} for the adversary.

We define negligible functions and probabilistic polynomial-time algorithms in the usual way; further details can be found in [1].

Identifiers. An identifier is a string of arbitrary length: $id \in \{0, 1\}^*$. A hierarchical identifier of depth d is a d -tuple of identifiers: $HID := (id_1, \dots, id_d) \in (\{0, 1\}^*)^d$, where the hierarchy descends from left to right. In other words, the ancestors of HID are $root := ()$, (id_1) , $(id_1, id_2), \dots$, and finally (id_1, \dots, id_{d-1}) , which is HID 's parent. Note that $root$ has depth $d = 0$. We denote the i -th ancestor of HID with $HID|_{-i}$, where $HID|_{-1}$ is the first ancestor or parent (id_1, \dots, id_{d-1}) of HID and $HID|_{-d}$ the d -th ancestor of HID , which is the root identifier $root$. Finally, we map integers to identifiers by using their binary representation, e.g., $id \leftarrow 5$ leads to $id = 101$ due to $(5)_2 = 101$.

3.3 Cryptographic Building Blocks

Pseudo-random Sequence (PRS). We formally define the term PRS in the technical report [1]. Informally, a PRS is a long bit string that no probabilistic polynomial-time algorithm can distinguish from a truly random bit string of identical length. In order to create a PRS, a Pseudo-Random Sequence Generator (PRSG), which is a deterministic polynomial-time algorithm G , derives the bit string from a random seed. As in [5,6,7,8] we define this derivation to occur sequentially: The output of G has the same length as its input, and the long bit string is produced by recursively applying G several times.

Hierarchical Identity Based Encryption. In short, an HIBE scheme allows to use the recipient's hierarchical identity as the public key of an asymmetric encryption scheme [16,17,18,19,10]. To set up the scheme, the center generates system parameters par and the secret master key MK using the key generation algorithm $(par, MK) \leftarrow SetupH(d_{max}, \lambda_H)$, where d_{max} is the maximum hierarchical depth (or tree height), λ_H a security parameter, and $MK = SK_{root}$

the secret key at the root. To derive the secret key SK_{HID} of a child node HID from its parent $HID|-1$, the parent uses the key generation algorithm $SK_{HID} \leftarrow \text{GenH}(HID, SK_{HID|-1})$. To encrypt message M exclusively for HID (and its ancestors $HID|-1, \dots, HID|-(d-1), \text{root}$), the center uses the encryption algorithm $C \leftarrow \text{EncH}(HID, M)$, where the output is the ciphertext C . Only HID (and its ancestors) has a matching secret key SK_{HID} that allows to decrypt C and obtain M using the decryption algorithm $M \leftarrow \text{DecH}(SK_{HID}, C)$.

Symmetric Broadcast Encryption. A symmetric BE scheme allows the center to efficiently encrypt a message intended for a large set of non-revoked receivers. To set up the scheme, the center generates the secret master key \mathcal{MK} using the key generation algorithm $\mathcal{MK} \leftarrow \text{GenB}(N, \lambda_B)$, where N is the number of receivers and λ_B a security parameter. To add receiver u_i to the system, the center uses the key extraction algorithm $SK_i \leftarrow \text{ExtrB}(\mathcal{MK}, i)$ to extract the secret key SK_i of u_i . To encrypt message M exclusively for the non-revoked receivers $\mathcal{U} \setminus \mathcal{R}$, the center uses the encryption algorithm $C \leftarrow \text{EncB}(\mathcal{MK}, \mathcal{R}, M)$, where the output is the ciphertext C . Only a non-revoked receiver u_i has a matching secret key SK_i that allows to decrypt C and obtain M using the decryption algorithm $M \leftarrow \text{DecB}(i, SK_i, C)$.

As mentioned in Sect. 2, the schemes [5,6,7,8] are *subset cover* BE schemes. In all such schemes, EncB covers any set of non-revoked receivers $\mathcal{U} \setminus \mathcal{R}$ using a well-chosen set $\{\mathcal{S}_j | j \in \mathcal{J}\} \subseteq \mathcal{P}(\mathcal{U})$ of subsets of \mathcal{U} . Specifically, each of the chosen subsets \mathcal{S}_j is related to a key, which ExtrB gives to all receivers within \mathcal{S}_j . To cover $\mathcal{U} \setminus \mathcal{R}$, EncB finds a disjoint union \mathcal{COV} of subsets containing only non-revoked receivers, such that all receivers in $\mathcal{U} \setminus \mathcal{R}$ are part of the cover and the cover size $|\mathcal{COV}|$ is minimal.

To achieve efficiency, the ciphertext C consists of three parts $C := (I, K, C')$, where I is the indexing information that describes \mathcal{COV} , $K := (\text{Enc}(dk_1, k), \dots, \text{Enc}(dk_{|\mathcal{COV}|}, k))$ a sequence of symmetric encryptions of the same session key k under all subset keys $dk_1, \dots, dk_{|\mathcal{COV}|}$ related to \mathcal{COV} , and $C' := \text{Enc}(k, M)$ the symmetric encryption of M under the session key [5,6,7,8]. The efficiency gain arises from the fact that M is a long message and k a short session key, avoiding several encryptions of the long message.

Derivation Broadcast Encryption. A symmetric BE scheme uses a generic symmetric encryption scheme as a building block. Center and receivers thus have to use the same key for encryption and decryption, respectively. The encryption algorithm EncB derives some symmetric keys dk_1, dk_2, \dots from \mathcal{MK} , such that each receiver in $\mathcal{U} \setminus \mathcal{R}$ can derive one such key from its secret key SK_i .

In short, a derivation BE scheme is a symmetric subset cover scheme in which $\mathcal{MK} := \{mk_j | j \in \mathcal{J}\}$ and $\mathcal{DK} := \{dk_l | l \in \mathcal{L}\}$ are finite sets of keys indexed by \mathcal{J} and \mathcal{L} such that each subset key dk_l is derived from exactly one element mk_j of \mathcal{MK} using a set of PRSGs. More formally, we define the scheme as follows:

Definition 1. Let $\mathcal{PRSG} := \{G_m | 1 \leq m \leq n\}$ be a finite set of distinct PRSGs. Let $d_{\max} \in \mathbb{N}$ be finite. Then the symmetric subset cover BE scheme \mathcal{SBE} is a derivation BE scheme if for all $dk_l \in \mathcal{DK}$ we have that $dk_l = G_{m_d} \circ \dots \circ G_{m_2}(mk_j)$

for exactly one master key element $mk_j \in \mathcal{MK}$ and a specific depth d , where $G_{m_2}, \dots, G_{m_d} \in \mathcal{PRSG}$ and $d \leq d_{\max}$.

As already pointed out in [5,7], we can replace the n distinct PRSGs with a single PRSG whose output is n times longer than its input. To calculate G_m , we then need to parse the output in order to obtain the m -th substring.

Based on Definition 1, we can represent all derived subset keys in \mathcal{DK} as the leaves of a forrest of trees that has (i) $|\mathcal{MK}|$ trees, (ii) one element mk_j of \mathcal{MK} at the root of each tree, (iii) maximum depth $d_{\max} - 1$, and (iv) at most n children per node. In addition, we can easily assign the index m of the corresponding PRSG as a tag to each edge of the trees. Setting $m_1 := j$, we can unambiguously assign the index $l := (m_1, m_2, \dots, m_d)$ to dk_l . In the example of Fig. 1, we have $dk_l = G_1 \circ G_n \circ G_1(mk_j)$ with $l = (j, 1, n, 1)$.

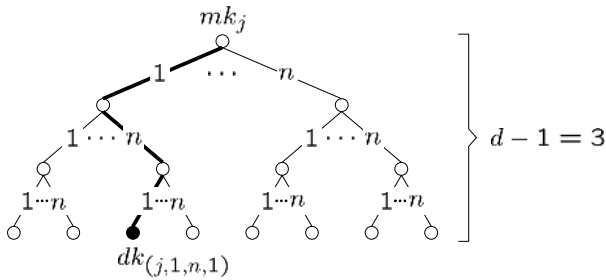


Fig. 1. Example of a key derivation tree for $d - 1 = 3$

4 Security Requirements

In this section we formalize the security requirement of a generic asymmetric BE scheme. Following common practice [20,15], we do not analyze the security of the symmetric encryption scheme used to encrypt message M under the session key k . Instead, we only analyze the security of the broadcast header (I, K) , which encapsulates k for the non-revoked receivers. In the technical report [1] we prove that this approach is valid as long as the symmetric encryption scheme is IND-CPA-secure. An interesting detail of the proof in [1] is the fact that this scheme only needs to withstand an IND-CPA attack in which the adversary does not have access to an encryption oracle. The security requirement on the symmetric encryption scheme is therefore quite moderate.

The algorithms of an asymmetric BE scheme are similar to those of a symmetric BE scheme. Compared to symmetric BE schemes as introduced in Sect. 3.3, an *asymmetric BE scheme* differs in two ways. First, the key generation algorithm has the public key PK as additional output: $(\mathcal{MK}, PK) \leftarrow \text{GenB}(N, \lambda_B)$. Second, the encryption algorithm uses the public key instead of the master key: $C \leftarrow \text{EncB}(PK, \mathcal{R}, k)$.

We require the asymmetric BE scheme generated by our transformation to provide the same security as the symmetric scheme from which it is derived. All

relevant symmetric schemes provide chosen ciphertext security of type IND-CCA1 [5,6,7,8], where the adversary may query its oracles only prior to the challenge.

More precisely, IND-CCA1 security of \mathcal{ABE} is defined under the following attack game between an adversary \mathcal{A} and a challenger \mathcal{C} : In the *setup phase*, \mathcal{C} uses $\text{GenB}(N, \lambda_{\mathcal{B}})$ to generate an instance of the scheme, initializes the set of revoked receivers as $\mathcal{R} = \emptyset$, and sends $(N, \lambda_{\mathcal{B}}, PK)$ to \mathcal{A} . The *find phase* begins, and \mathcal{A} may send adaptive key extraction and decryption queries to an oracle. A key extraction query asks for the secret key \mathcal{SK}_i of a specific receiver u_i , which \mathcal{C} then adds to \mathcal{R} . A decryption query (i, I, K) asks for decryption of a broadcast header (I, K) chosen by \mathcal{A} , where the decryption oracle plays the role of a specific non-revoked receiver u_i that executes DecB . When \mathcal{A} believes to have gathered enough information to mount a successful attack, it tells \mathcal{C} that the find phase is over. Then \mathcal{C} generates two independent random session keys k_0 and k_1 , tosses a coin $b \xleftarrow{R} \{0, 1\}$ and executes $(I, K_b) \leftarrow \text{EncB}(PK, \mathcal{R}, k_b)$, where \mathcal{R} is the final set of revoked receivers after all key extraction queries. In the *guess phase*, \mathcal{A} needs to guess b from the challenge $k_0, k_1, (I, K_b)$ and wins if the guess b' equals b . We define the IND-CCA1 advantage of \mathcal{A} against \mathcal{ABE} as $\text{Adv}_{\mathcal{ABE}, \mathcal{A}}^{\text{ind-cca1}}(\lambda_{\mathcal{B}}) := \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]$, where the randomness is taken over all coin tosses of \mathcal{C} , \mathcal{A} , GenB , and EncB .

Definition 2. *The scheme \mathcal{ABE} is $(n_{\mathcal{X}}, n_{\mathcal{D}})$ -IND-CCA1-secure if for any probabilistic polynomial-time adversary \mathcal{A} that makes at most $n_{\mathcal{X}}$ key extraction queries and at most $n_{\mathcal{D}}$ decryption queries in the above game, we have that $\text{Adv}_{\mathcal{ABE}, \mathcal{A}}^{\text{ind-cca1}}(\lambda_{\mathcal{B}})$ is a negligible function.*

We will reduce the security of our asymmetric BE scheme to that of the HIBE scheme with which it is instantiated. We therefore need to define IND-CCA1 security of HIBE. As the attack game is analogous to that of a BE scheme, we only describe the differences. In the find phase, \mathcal{A} also performs adaptive key extraction and decryption queries, but obviously queries the oracle with a specific hierarchical identifier HID instead of a specific receiver. \mathcal{C} adds HID to the set \mathcal{R}_H of revoked hierarchical identifiers. At the end of the find phase, \mathcal{A} chooses two message M_0 and M_1 as well as a hierarchical identifier HID^* on which it wishes to be challenged, where no ancestor of HID^* may be in \mathcal{R}_H . The challenge is $C_b := \text{EncH}(HID^*, M_b)$, where $b \xleftarrow{R} \{0, 1\}$ is a coin that \mathcal{C} tosses.

5 Proposed Solution

5.1 Overview

The transformation from the symmetric to the asymmetric setting generically uses an HIBE scheme in order to translate the derivation pattern of the symmetric scheme to that of an asymmetric scheme. The major task is to find an efficient mapping between the derivation pattern and a hierarchical identifier of the HIBE scheme. To define this mapping, we reuse the indices of the PRSGs used in the derivation BE scheme and interpret them as identifiers in a hierarchical identifier. The hierarchical identifier starts with the index of the master

key from which the derived key originates. The remaining elements of the hierarchical identifier are the indices of the PRSGs used to compute the derived key. With this mapping, the center and the receivers can compute all derived (private) keys in the asymmetric setting just like they would in the symmetric setting. A receiver obtains the keys at the same position in the HIBE hierarchy as in the symmetric key derivation tree. The application of the HIBE key generation algorithm replaces the application of a PRSG. All other algorithms of the symmetric BE scheme transfer to the asymmetric setting. Specifically, the algorithms for finding a cover, structuring the ciphertext tuple $C = (I, K, C')$, and finding a receiver's subset in the cover remain unchanged.

5.2 Details on the Transformation to the Asymmetric Setting

In this section we specify the mapping between the derivation pattern in Definition 1 and the hierarchical identifiers used in the generic HIBE scheme. Let $mk_j \in \mathcal{MK}$ and $dk_l \in \mathcal{DK}$ be any pair of a master key and a key derived from this master key. Further, let dk_l follow the derivation pattern $dk_l = G_{m_d} \circ \dots \circ G_{m_2}(mk_j)$ of Definition 1. Then we define $m_1 := j$, which is the index of mk_j , and set the hierarchical identifier of depth d to be

$$HID := (id_1, id_2, \dots, id_d) \leftarrow (m_1, m_2, \dots, m_d)$$

In the asymmetric setting, the key generation algorithm **GenB** first generates an instance of an HIBE scheme using **SetupH**. Then it generates the asymmetric master key elements $mk_{(id_1)} \in \mathcal{MK}$ by deriving them from the HIBE master key, which serves as the root:

$$mk_{(id_1)} \leftarrow \text{GenH}((id_1), MK) \quad \text{with} \quad MK = mk_{\text{root}}$$

All derived keys follow the same derivation pattern as in the symmetric setting, but the application of the HIBE key generation algorithm replaces the application of the PRSG:

$$dk_{(m_1, \dots, m_d)} \leftarrow G_{m_d}(dk_{(m_1, \dots, m_{d-1})}) \quad \text{becomes} \quad dk_{HID} \leftarrow \text{GenH}(HID, dk_{HID|_{-1}})$$

Note that the first derivation $dk_{(id_1, id_2)} \leftarrow \text{GenH}((id_1, id_2), dk_{(id_1, id_2)|_{-1}})$ starts at the master key element $dk_{(id_1, id_2)|_{-1}} = dk_{(id_1)} := mk_{(id_1)}$. Assembling all derived keys, we obtain a tree structure in which **root** represents a common root for the forrest of trees of the derivation BE scheme. The only other difference to the symmetric setting is that HIBE encryptions (and decryptions) replace the symmetric encryptions of the session key:

$$\text{Enc}(dk_{(m_1, \dots, m_d)}, k) \quad \text{becomes} \quad \text{EnCH}(HID, k)$$

As a special case for depth $d = 1$, the HIBE scheme becomes an identity based encryption (IBE) scheme without hierarchy. Thereby we can transform any *improper* derivation BE scheme to the asymmetric setting, i.e., any scheme that only uses the master keys, but does not derive any keys, resulting in $\mathcal{MK} = \mathcal{DK}$. Dodis and Fazio [9] have also used an IBE scheme to transform the CS scheme, which is an improper derivation BE scheme. However, their solution is individual and not part of a generic framework as in our proposed solution.

5.3 Security Analysis

We summarize the security of our proposed framework in the following theorem:

Theorem 1. *Let SBE be an IND-CCA1-secure derivation BE scheme. Let ABE be the asymmetric BE scheme generated by our framework. Let $HIBE$ be the $(|\mathcal{SK}_i| \cdot n_X, n_D)$ -IND-CCA1-secure HIBE scheme of ABE . Then ABE is (n_X, n_D) -IND-CCA1-secure.*

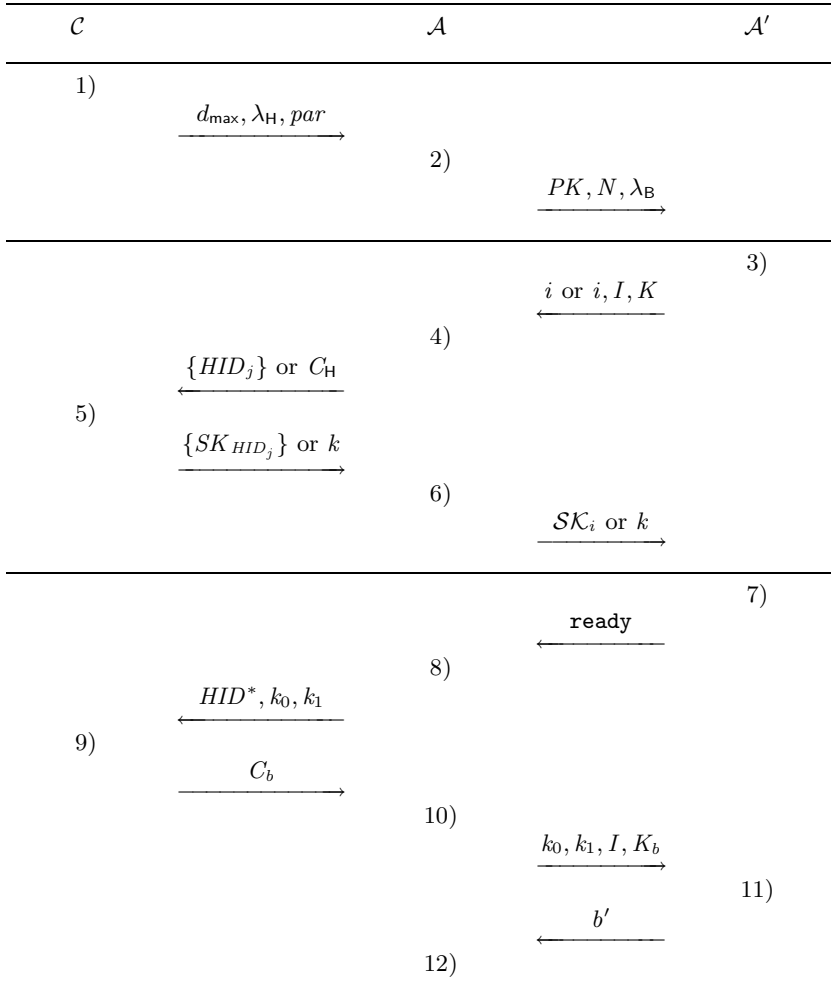


Fig. 2. Construction of $HIBE$ adversary \mathcal{A} based on ABE adversary \mathcal{A}'

Proof. (SKETCH) The proof is by contradiction. Let there be an IND-CCA1 adversary \mathcal{A}' against ABE such that the advantage $\text{Adv}_{ABE, \mathcal{A}'}^{\text{ind-cca1}}(\lambda_B)$ is not negligible.

By using \mathcal{A}' as a subroutine, we construct an IND-CCA1 adversary \mathcal{A} against \mathcal{HIBE} with an advantage $\text{Adv}_{\mathcal{HIBE}, \mathcal{A}}^{\text{ind-cca1}}(\lambda_{\mathcal{H}})$ that is not negligible, which contradicts the IND-CCA1 security of \mathcal{HIBE} and concludes the proof. We show each step of \mathcal{A} 's construction in Fig. 2:

1. The HIBE challenger \mathcal{C} sets up the HIBE scheme $(par, MK) \leftarrow \text{SetupH}(d_{\max}, \lambda_{\mathcal{H}})$, initializes $\mathcal{R}_{\mathcal{H}} \leftarrow \emptyset$, and sends $(d_{\max}, \lambda_{\mathcal{H}}, par)$ to \mathcal{A} .
2. \mathcal{A} determines the constant-size public key PK of the asymmetric BE scheme, which is simply the rule for determining HID using the mapping between symmetric and asymmetric setting. Then \mathcal{A} sends $(PK, N, \lambda_{\mathcal{B}})$ to \mathcal{A}' .
3. \mathcal{A}' performs key extraction and decryption queries in any adaptively chosen order (Steps 3 to 6). In a key extraction query, \mathcal{A}' selects the index i of a specific receiver u_i . In a decryption query, \mathcal{A}' selects the index i of a specific receiver u_i and a broadcast header (I, K) . \mathcal{A}' sends the query to \mathcal{A} , which simulates the answers of the oracle that \mathcal{A}' calls.
4. In a key extraction query, \mathcal{A} derives the hierarchical identifiers $\{HID_j | j \in \mathcal{J}\}$ of all subset keys in \mathcal{SK}_i using the derivation BE scheme and the mapping, thus introducing the linear factor $|\mathcal{SK}_i|$ of $n_{\mathcal{X}}$. Then \mathcal{A} sends a key extraction query for each element of $\{HID_j | j \in \mathcal{J}\}$ to \mathcal{C} . Finally, \mathcal{A} adds u_i to \mathcal{R} . In a decryption query, \mathcal{A} parses K into $|\mathcal{COV}|$ encryptions of an unknown session key k . Then \mathcal{A} uses I to extract from K the encryption $C_{\mathcal{H}} = \text{EncH}(HID_j, k)$ that corresponds with the subset \mathcal{S}_j that covers u_i . Finally, \mathcal{A} sends $C_{\mathcal{H}}$ to \mathcal{C} .
5. \mathcal{C} responds with the answers of the two oracles, i.e., $\{SK_{HID_j} | j \in \mathcal{J}\}$ or k . For each key extraction query, \mathcal{C} adds $\{HID_j | j \in \mathcal{J}\}$ to the set $\mathcal{R}_{\mathcal{H}}$ of revoked hierarchical identifiers. As the two oracles know MK , they answer all queries correctly.
6. For a key extraction query, \mathcal{A} assembles the $|\mathcal{SK}_i|$ private HIBE keys into the secret key SK_i of u_i . For a decryption query, \mathcal{A} sends k to \mathcal{A}' .
7. \mathcal{A}' tells \mathcal{A} that the find phase is over.
8. \mathcal{A} calculates the cover $\mathcal{COV} := \{\mathcal{S}_1, \dots, \mathcal{S}_{|\mathcal{COV}|}\}$ for the final non-revoked receivers $\mathcal{U} \setminus \mathcal{R}$ and chooses one subset $\mathcal{S}_{j^*} \in \mathcal{COV}$ at random. Then it finds the hierarchical identifier $HID^* := HID_{j^*}$ that corresponds with \mathcal{S}_{j^*} . Finally, it generates two random session keys k_0 and k_1 and sends (k_0, k_1, HID^*) to \mathcal{C} . Note that HID^* cannot be in $\mathcal{R}_{\mathcal{H}}$ as $\mathcal{S}_{j^*} \in \mathcal{COV}$ and thus $\mathcal{S}_{j^*} \cap \mathcal{R} = \emptyset$, whereas $\mathcal{R}_{\mathcal{H}}$ contains hierarchical identifiers of subsets with at least one revoked receiver. The same holds for all ancestors of HID^* : If any of them was in $\mathcal{R}_{\mathcal{H}}$, then the scheme \mathcal{SBE} could not be IND-CCA1-secure, as revoked receivers could derive a key in the cover.
9. \mathcal{C} tosses a coin $b \xleftarrow{\mathcal{R}} \{0, 1\}$ and sends challenge $C_b = \text{EncH}(HID^*, k_b)$ to \mathcal{A} .
10. For all subsets $\mathcal{S}_j \in \mathcal{COV}$, \mathcal{A} constructs a session key encryption in the following way: For the first $j^* - 1$ subsets in the cover, it encrypts k_1 . For the j^* -th subset, it uses the challenge C_b . For the remaining $|\mathcal{COV}| - j^*$ subsets, it encrypts k_0 . Let HID_j correspond with \mathcal{S}_j according to the mapping. Then the challenge from \mathcal{A} to \mathcal{A}' is (k_0, k_1, I, K_b) with $K_b := (\text{EncH}(HID_1, k_1), \dots, \text{EncH}(HID_{j^*-1}, k_1), C_b, \text{EncH}(HID_{j^*+1}, k_0), \dots, \text{EncH}(HID_{|\mathcal{COV}|}, k_0))$

11. \mathcal{A}' outputs the guess b' and sends it to \mathcal{A} .
12. \mathcal{A} forwards b' to \mathcal{C} .

We need to prove that $\text{Adv}_{\mathcal{HIBE}, \mathcal{A}}^{\text{ind-ccal}}(\lambda_{\mathcal{H}})$ is not negligible. The proof uses a standard hybrid argument. The hybrid experiments are identical to the regular attack game against \mathcal{ABE} except for the session key encryptions in the challenge. In the hybrid experiment indexed $j \in \{0, 1, \dots, |\mathcal{COV}|\}$, \mathcal{A}' obtains the following session key encryptions: $(\text{EncH}(\text{HID}_1, k_1), \dots, \text{EncH}(\text{HID}_j, k_1), \text{EncH}(\text{HID}_{j+1}, k_0), \dots, \text{EncH}(\text{HID}_{|\mathcal{COV}|}, k_0))$. Note that the first j session key encryptions involve k_1 and the last $|\mathcal{COV}| - j$ such encryptions involve k_0 . Let $\text{Pr}_j[b' = 1]$ denote the probability that \mathcal{A}' outputs $b' = 1$ in the hybrid experiment indexed j . For $j = 0$ \mathcal{A}' obtains the same broadcast header as in the regular attack game against \mathcal{ABE} for $b = 0$. Conversely, \mathcal{A}' obtains the same broadcast header for $j = |\mathcal{COV}|$ in the hybrid experiment and $b = 1$ in the regular attack game:

$$\begin{aligned} \text{Adv}_{\mathcal{ABE}, \mathcal{A}'}^{\text{ind-ccal}}(\lambda_{\mathcal{B}}) &= \text{Pr}[b' = 1|b = 1] - \text{Pr}[b' = 1|b = 0] \\ &= \text{Pr}_{|\mathcal{COV}|}[b' = 1] - \text{Pr}_0[b' = 1] = \sum_{j=1}^{|\mathcal{COV}|} \text{Pr}_j[b' = 1] - \text{Pr}_{j-1}[b' = 1], \end{aligned}$$

where the last equality adds some intermediate terms that add to zero. As our constructed adversary \mathcal{A} uses \mathcal{A}' as a subroutine, we can express its advantage using the same hybrid experiments. Note that for $j^* = j$ and $b = 1$, \mathcal{A}' obtains from \mathcal{A} the same input as in hybrid experiment j , whereas $j^* = j$ and $b = 0$ corresponds with hybrid experiment $j-1$:

$$\begin{aligned} \text{Adv}_{\mathcal{HIBE}, \mathcal{A}}^{\text{ind-ccal}}(\lambda_{\mathcal{H}}) &= \text{Pr}[b' = 1|b = 1] - \text{Pr}[b' = 1|b = 0] \\ &= \sum_{j=1}^{|\mathcal{COV}|} \left[(\text{Pr}[b' = 1|b = 1, j^* = j] - \text{Pr}[b' = 1|b = 0, j^* = j]) \cdot \text{Pr}[j^* = j] \right] \\ &= \frac{1}{|\mathcal{COV}|} \sum_{j=1}^{|\mathcal{COV}|} \text{Pr}_j[b' = 1] - \text{Pr}_{j-1}[b' = 1], \end{aligned}$$

where the last equality uses the fact that j^* is selected with uniform distribution. For our constructed adversary \mathcal{A} we therefore obtain $\text{Adv}_{\mathcal{HIBE}, \mathcal{A}}^{\text{ind-ccal}}(\lambda_{\mathcal{H}}) = 1/|\mathcal{COV}| \cdot \text{Adv}_{\mathcal{ABE}, \mathcal{A}'}^{\text{ind-ccal}}(\lambda_{\mathcal{B}})$, which is not negligible because $\text{Adv}_{\mathcal{ABE}, \mathcal{A}'}^{\text{ind-ccal}}(\lambda_{\mathcal{B}})$ is not negligible. This concludes the proof. The full version of this proof is available in [1].

6 Two Exemplary Instantiations

In the sequel, we apply our framework to two exemplary derivation BE schemes as proof of concept. The first scheme is the SD scheme of [5], where our solution is semantically identical to that of Dodis and Fazio in [9]. The second scheme is the LPI scheme of [7], which has a smaller receiver storage size than SD and for which no asymmetric version exists to the best of our knowledge.

6.1 Asymmetric Subset Difference Scheme

Overview of the Subset Difference Scheme. In this section we describe only those parts of the original symmetric SD scheme that are required in order

to understand our notation. For further details we refer to the original paper [5]. The key generation algorithm **GenB** assigns the N receivers to the leaves of a binary tree. Without loss of generality, we assume that the tree is balanced and its height $\log_2 N$ an integer; otherwise the height is rounded up to $\lceil \log_2 N \rceil$. **GenB** assigns to each non-leaf node v_i a λ_B -bit string $label_i$, which it chooses at random with independent uniform distribution.

The key extraction algorithm **ExtrB** uses $n = 3$ distinct PRSGs $G_1, G_2,$ and G_3 . For each label $label_i$, **ExtrB** derives a label for all children of node v_i in the following way: For the left child, the derived label is $G_1(label_i)$, and analogously $G_2(label_i)$ for the right child. Let $v_{j'}$ be a child of node v_j and let $m = 1$ for the left and $m = 2$ for the right child. Then all derived labels follow a derivation pattern starting with $label_{i,i} := label_i$:

$$label_{i,j'} \leftarrow G_m(label_{i,j}) \quad \text{with } m \in \{1, 2\} \quad \text{for left and right child} \quad (1)$$

Each label $label_{i,j}$ represents a specific subset $\mathcal{S}_{i,j}$ of receivers. Specifically, $\mathcal{S}_{i,j}$ contains all receivers that are below node v_i , but not below v_j . Each receiver obtains the labels of all subsets to which it belongs. However, the authors of [5] ingeniously chose the derivation pattern (1) such that a receiver needs to store only a small fraction of its labels and can still derive all of them. Specifically, each receiver actually stores only the labels $label_{i,j}$ that are just one node off the path from the receiver's leaf to the root (for an example of these labels, see the black solid dots on the left-hand side of Fig. 3).

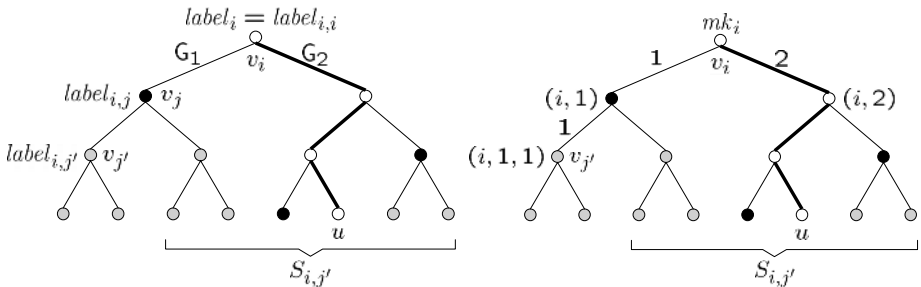


Fig. 3. Left: The three labels indicated with black solid dots are one node off the path from receiver u to the root (black thick line). By storing them, receiver u can derive all labels indicated with grey shaded dots. Right: By labeling the tree edges with the PRSG index, we can read the hierarchical identifier from the path.

The encryption algorithm **EncB** finds a cover \mathcal{COV} of minimal size. The authors of [5] showed that the cover size is $|\mathcal{COV}| = 2 \cdot |\mathcal{R}| - 1$ in the worst case. For each subset $\mathcal{S}_{i,j} \in \mathcal{COV}$, **EncB** calculates $k_{i,j} \leftarrow G_3(label_{i,j})$ and encrypts the session key k with $k_{i,j}$. The indexing information I describes the cover \mathcal{COV} .

The decryption algorithm **DecB** executed by a non-revoked receiver $u \in \mathcal{U} \setminus \mathcal{R}$ uses I to find the subset $\mathcal{S}_{i,j'}$ in the cover to which u belongs. It decrypts the session key using $k_{i,j'}$, which it derives from a stored label $label_{i,j}$ such that v_j is an ancestor of $v_{j'}$.

Mapping to Our Notation for Derivation BE. The following mapping describes how we map each key in the original scheme to our notation of a derivation BE scheme. The master keys mk_i are the labels $label_{i,i} = label_i$. The set of PRSGs is $\mathcal{PRSG} = \{G_1, G_2, G_3\}$. Center and receivers derive the keys dk_l in the following way, which is of the form $dk_l = G_{m_d} \circ \dots \circ G_{m_2}(mk_j)$ from Definition 1. Let $label_{i,j}$ be a $(d - 2)$ -th descendant of master key $mk_i = label_i$, where the two PRSGs G_1 and G_2 have been applied in the following order: $label_{i,j} = G_{m_{d-1}} \circ \dots \circ G_{m_2}(mk_i)$ with $m_\alpha \in \{1, 2\}$ according to (1). Then the derived key is $dk_l = G_3 \circ G_{m_{d-1}} \circ \dots \circ G_{m_2}(mk_i)$, which completes the mapping. For example, by comparing left and right-hand side of Fig. 3, we obtain $k_{i,j'} = G_3(label_{i,j'}) = G_3 \circ G_1 \circ G_1(mk_i) = dk_{(i,1,1,3)}$, which u can calculate.

6.2 Asymmetric Layered Punctured Interval Scheme

Overview of the Punctured Interval Scheme. Again we only describe the parts relevant for our notation. As the original symmetric LPI scheme is more involved than SD, we explain it incrementally as in [7]. First, we define punctured intervals. Second, we explain the punctured interval scheme without layers, while we add the layers in the technical report [1] due to lack of space.

Let the N receivers be N nodes on a straight line from left to right, with u_1 being the left-most and u_N being the right-most receiver. A p -punctured c -interval is a subset $\mathcal{S} \subseteq \mathcal{U}$ of receivers that contains c or less consecutive receivers starting from and ending at non-revoked receivers and containing p or less revoked receivers. More specifically, $\mathcal{S}_{i,j;r_1,\dots,r_q}$ is the p -punctured c -interval that starts at node u_i , ends at node u_j , and contains the revoked receivers r_1, \dots, r_q , where $1 \leq j - i + 1 \leq c$, $0 \leq q \leq p$, and $i < r_1 < \dots < r_q < j$. We give an example in Fig. 4, where black solid dots represent revoked receivers.

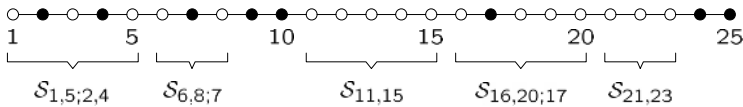


Fig. 4. From left to right, five 2-punctured 5-intervals contain all non-revoked receivers

For the p -punctured c -interval scheme $(p;c)\text{-}\pi$, we explain its four algorithms (GenB, ExtrB, EncB, DecB) as introduced in Sect. 3.3. GenB generates N master keys of length λ_B , which we denote $k_{1,1}, k_{2,2}, \dots, k_{N,N}$. The key extraction algorithm ExtrB extracts all keys intended for receiver u_i , which are the keys related to all p -punctured c -intervals that contain u_i as a non-revoked receiver. To do so, ExtrB derives several keys from each master key. Specifically, each master key $k_{i,i}$ is at the root of a key tree in which each leaf represents one p -punctured c -interval that starts at u_i . Each non-revoked receiver is on the path from root to leaf, while the revoked receivers are not on the path. The keys of non-revoked receivers indicate their relative position in the tree as shown in Fig. 5, where each row represents one p -punctured 5-interval for $0 \leq p \leq 3$. Similar to the

p -punctured c -interval $\mathcal{S}_{i,j;r_1,\dots,r_q}$, the derived keys $k_{i,j;r_1,\dots,r_q}$ indicate the start node u_i (and thus the master key $k_{i,i}$), the recipient u_j of this particular key, and the revoked receivers r_1, \dots, r_q between them, which obtain no key. For example, the key $k_{1,5;2,4}$ in Fig. 5 is derived from $k_{1,1}$, given to receiver u_5 , and excludes u_2 as well as u_4 . Note that the corresponding 2-punctured 5-interval $\mathcal{S}_{1,5;2,4}$ appears both on the left of Fig. 4 and as the fifth row of Fig. 5.

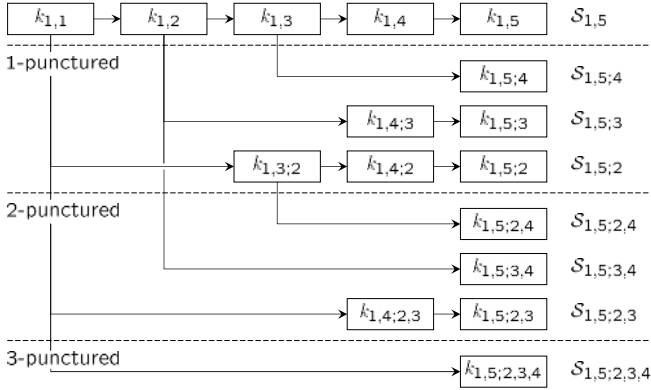


Fig. 5. Key tree starting at root $k_{1,1}$ (left) and ending in leaves (right); each row is a p -punctured 5-interval

It remains to define the PRSGs that actually derive the keys $k_{i,j;r_1,\dots,r_q}$ from $k_{i,i}$. Let $\mathcal{PRSG} := \{G_1, G_2, \dots, G_{p+1}\}$ be $p + 1$ distinct PRSGs. The master key $k_{i,i}$ corresponds with a non-revoked node u_i . For all non-revoked nodes further to the right, the following rule applies: Let $m \in \{1, 2, \dots, p+1\}$ be the distance between the two nearest non-revoked receivers u_{j_1} and u_{j_2} such that $m = j_2 - j_1 > 0$. Then PRSG G_m derives the key of u_{j_2} from that of u_{j_1} :

$$k_{i,j_2;r_1,\dots,r_{q+m-1}} \leftarrow G_m(k_{i,j_1;r_1,\dots,r_q}) \quad \text{with } j_2 = j_1 + m \text{ and } 1 \leq m \leq p+1 \quad (2)$$

We give five examples from Fig. 5 in order to illustrate this derivation pattern: $k_{1,2} \leftarrow G_1(k_{1,1})$, $k_{1,4;3} \leftarrow G_2(k_{1,2})$, $k_{1,5;2,4} \leftarrow G_2(k_{1,3;2})$, $k_{1,5;3,4} \leftarrow G_3(k_{1,2})$, and $k_{1,5;2,3,4} \leftarrow G_4(k_{1,1})$

The encryption algorithm EncB finds a minimum-size cover for the non-revoked receivers, such that each of them is contained in one p -punctured c -interval (Fig. 4 is an example). For each such interval, EncB encrypts the session key k with the last key of the corresponding key chain. The decryption algorithm DecB executed by a non-revoked receiver uses the indexing information I to find the p -punctured c -interval in which it is covered. Then it derives the last key of this key chain, decrypts the session key, and subsequently the message.

Mapping to Our Notation for Derivation BE. The following mapping describes how each key in the punctured interval scheme is mapped to our notation

of a derivation BE scheme (see Definition 1). The master keys mk_i are the keys $k_{i,i}$. \mathcal{PRSG} is the set of PRSGs. Center and receivers derive the keys dk_l in the following way, which is of the required form $dk_l = G_{m_d} \circ \dots \circ G_{m_2}(mk_j)$. Let $k_{i,j;r_1,\dots,r_q}$ be a descendant of master key $mk_i = k_{i,i}$. By tracing back the path from leaf to root as in Fig. 5, we find the indices of the PRSGs that were used to derive $k_{i,j;r_1,\dots,r_q}$ by using (2) repeatedly. Let the order of indices be as follows: $k_{i,j;r_1,\dots,r_q} = G_{m_d} \circ \dots \circ G_{m_2}(k_{i,i})$ with $m_\alpha \in \{1, 2, \dots, p + 1\}$ according to (2). Then this representation already complies with Definition 1, which completes the mapping. We cover the full LPI scheme in the technical report [1].

7 Conclusion

In this paper we have proposed a generic framework that allows to transform a large class of secure symmetric Broadcast Encryption (BE) schemes into secure asymmetric BE schemes. We have proven the framework to be as secure as the hierarchical identity based encryption scheme with which it is instantiated. We have given two examples of practically relevant BE schemes that fall within our framework, the second of which hadn't been solved so far.

References

1. Huber, U., Sadeghi, A.R.: A generic transformation from symmetric to asymmetric broadcast encryption. Technical Report, Horst Görtz Institute for IT Security (2006) <http://www.prosec.rub.de/publications>.
2. Wallner, D.M., Harder, E.J., Agee, R.C.: Key management for multicast: Issues and architectures. Request for comments (2627), Internet Engineering Task Force (IETF) (1999) URL [ftp.ietf.org/rfc/rfc2627.txt](ftp://ftp.ietf.org/rfc/rfc2627.txt).
3. 4C Entity, LLC: CPPM specification—introduction and common cryptographic elements. Specification Revision 1.0 (2003) URL <http://www.4centity.com/data/tech/spec/cppm-base100.pdf>.
4. AACS Licensing Administrator: Advanced access content system (AACS): Introduction and common cryptographic elements. Specification Revision 0.90 (2005) URL http://www.aacsla.com/specifications/AACS_Spec-Common_0.90.pdf.
5. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In Kilian, J., ed.: CRYPTO 2001. Volume 2139 of LNCS, Springer (2001) 41–62
6. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In Yung, M., ed.: CRYPTO 2002. Volume 2442 of LNCS, Springer (2002) 47–60
7. Jho, N.S., Hwang, J.Y., Cheon, J.H., Kim, M.H., Lee, D.H., Yoo, E.S.: One-way chain based broadcast encryption schemes. [22] 559–574
8. Jho, N.S., Yoo, E.S., Cheon, J.H., Kim, M.H.: New broadcast encryption scheme using tree-based circle. In: ACM DRM 2005, ACM Press (2005) 37–44
9. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In Feigenbaum, J., ed.: Digital Rights Management Workshop. Volume 2696 of LNCS, Springer (2003) 61–80
10. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. [22] 440–456

11. Fiat, A., Naor, M.: Broadcast encryption. In Stinson, D.R., ed.: CRYPTO 1993. Volume 773 of LNCS, Springer (1994) 480–491
12. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In Frankel, Y., ed.: Financial Cryptography 2000. Volume 1962 of LNCS, Springer (2001) 1–20
13. Tzeng, W.G., Tzeng, Z.J.: A public-key traitor tracing scheme with revocation using dynamic shares. In Kim, K., ed.: Public Key Cryptography 2001. Volume 1992 of LNCS, Springer (2001) 207–224
14. Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Desmedt, Y., ed.: Public Key Cryptography 2003. Volume 2567 of LNCS, Springer (2002) 100–115
15. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In Shoup, V., ed.: CRYPTO 2005. Volume 3621 of LNCS, Springer (2005) 258–275
16. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In Zheng, Y., ed.: ASIACRYPT 2002. Volume 2501 of LNCS, Springer (2002) 548–566
17. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In Knudsen, L.R., ed.: EUROCRYPT 2002. Volume 2332 of LNCS, Springer (2002) 466–481
18. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. [21] 207–222
19. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. [21] 223–238
20. Attrapadung, N., Kobara, K., Imai, H.: Sequential key derivation patterns for broadcast encryption and key predistribution schemes. In Lai, C.S., ed.: ASIACRYPT 2003. Volume 2894 of LNCS, Springer (2003) 374–391
21. Cachin, C., Camenisch, J., eds.: Advances in Cryptology—EUROCRYPT 2004, Interlaken, Switzerland, May 2–6, 2004. Volume 3027 of LNCS, Springer (2004)
22. Cramer, R., ed.: Advances in Cryptology—EUROCRYPT 2005, Aarhus, Denmark, May 22–26, 2005. Volume 3494 of LNCS, Springer (2005)

Transparent Image Encryption Using Progressive JPEG*

Thomas Stütz and Andreas Uhl

Department of Computer Sciences
Salzburg University, Austria
{tstuetz, uhl}@cosy.sbg.ac.at

Abstract. Many application scenarios do not demand confidential encryption of visual data, but on the contrary require that certain image information is public (transparent encryption). One scenario is e.g., Pay-TV, where a low quality version should become public to attract possible customers. Transparent encryption can be implemented most efficiently in case of scalable bitstreams by encrypting enhancement layer data and baseline JPEG is therefore not well suited for designing such encryption schemes in an efficient manner. This paper investigates how transparent encryption can be realized through selective encryption of the progressive JPEG modes. The traditional approach which encrypts enhancement layers starting at the end of the bitstream suffers from high computational load. Encryption schemes with significantly reduced encryption effort are shown to deliver equivalent image quality and security.

1 Introduction

Encryption schemes for multimedia data need to be specifically designed to protect multimedia content and fulfil the application requirements for a particular multimedia environment [17].

For example, real-time encryption of visual data using classical ciphers requires heavy computation due to the large amounts of data involved, but many multimedia applications require security on a much lower level (e.g. TV news broadcasting [11]). In this context, several selective or partial encryption schemes have been proposed recently which do not strive for maximum security, but trade off security for computational complexity by restricting the encryption to the perceptually most relevant parts of the data.

However, encryption may have an entirely different aim as opposed to pure confidentiality in the context of multimedia applications. Macq and Quisquater [10, 11] introduce the term “transparent encryption” mainly in the context of digital TV broadcasting: a broadcaster of pay TV does not always intend to prevent unauthorised viewers from receiving and watching his program, but rather

* The work described in this paper is partially supported by the Austrian Science Fund, project no. 15170 and by the Austrian Grid Project, funded by the Austrian BMBWK (Federal Ministry for Education, Science and Culture) under contract GZ 4003/2-VI/4c/2004.

intends to promote a contract with nonpaying watchers. This can be facilitated by providing a low quality version of the broadcasted program for everyone, only legitimate (paying) users get access to the full quality visual data. This is meant also by the term “try and buy” scenario. Therefore, privacy is not the primary concern in such an environment. The simplest approach to achieve this would be to simply distribute both versions, a low quality version to all potential viewers, and a high quality version only to paying viewers. However, this is mostly not desired due to the excessive demand of storage and bandwidth.

Transparent encryption usually transmits a high quality version of the visual data to all possible viewers but aims at protecting the details of the data which enable a pleasant viewing experience in an efficient manner. If this data are missing, the user is (hopefully) motivated to pay for the rest of the data which may be accessed upon transmission of the required key material by the broadcaster. Another application area of transparent encryption are preview images in image and video databases. Therefore, there are two major requirements that have to be met concurrently:

- To hide a specific amount of image information (security requirement).
- To show a specific amount of image information (quality requirement).

While the first requirement is a generalization of the confidentiality encryption approach – the condition of full encryption of all image information is extended to a “specific amount” –, the second requirement, namely to explicitly demand a certain image quality, is completely different from scenarios where confidentiality or privacy are the primary aims.

To implement transparent encryption, Macq and Quisquater [11] propose to use line permutations in the transform domain of a lossless multiresolution transform. The permutations are only applied in the region of the transform domain corresponding to fine grained details of the data. Droogenbroeck and Benedett [6] propose to encrypt bitplanes of the binary representation of raw image data, contrasting to the privacy focused approach they suggest to start with the LSB bitplane. With respect to JPEG encoded images, the authors suggest to encrypt sign and magnitude bits of medium and high frequency DCT coefficients (note that this is again exactly just the other way round as compared to encrypting low frequency coefficients only for privacy protection [2, 9]). Droogenbroeck [5] extends this latter idea to “multiple encryption” where different sets of DCT coefficients are encrypted by different content owners, and “over encryption” where these sets do not have an empty intersection (i.e. coefficients are encrypted twice or even more often). Bodo et al. [1] propose a technique called “waterscrambling” where they embed a watermark into the motion vectors of an MPEG stream, thereby reducing the video quality significantly – only a legitimate user has access to the key and may descramble the motion vectors.

Transparent encryption may be implemented in the simplest way in the context of scalable or embedded bitstreams. Transparent encryption is achieved in this environment by simply encrypting the enhancement layer(s). This has been proposed by Kunkelmann and Horn using a scalable video codec based on a spatial resolution pyramid [9, 8] and by Dittmann and Steinmetz [3, 4] using a SNR

scalable MPEG-2 encoder/decoder. Yuan et al. [19] propose to use MPEG-4 FGS for transparent encryption, JPEG2000 transparent encryption is discussed in our earlier work [16].

The baseline JPEG format does not fit well into the transparent encryption scenario. For example, in order to selectively protect high frequency AC coefficients of a JPEG image (as discussed for example by Droogenbroeck and Benedett [6]), the file needs to be parsed for the EOB symbols 0x00 to identify the end of a 8×8 pixels block where the VLC codewords corresponding to these coefficients will be located (with two exceptions: if 0xFF is followed by 0x00, 0x00 is used as a stuffbit and has to be ignored and if AC63 (the last AC-Coefficient) does not equal 0 there will be no 0x00 and the AC coefficients have to be counted). It is clear that transparent encryption will be fairly inefficient under these circumstances where a significant parsing overhead is introduced.

In this work we systematically investigate the different JPEG progressive modes as defined in the JPEG extended system [13] with respect to their usefulness for providing efficient and yet secure transparent encryption schemes. Section 2 reviews the three modes which are also compared to the JPEG baseline system in terms of compression performance and data organisation. Section 3 finally discusses the respective suitability in a transparent encryption context, the paper is concluded in Section 4.

2 Progressive JPEG Modes

The basic idea of JPEG-based progressive coding [7, 15] is to organize the data into a base layer which contains a low quality approximation to the original data and several enhancement layers which, if combined with the base layer, successively improve the quality.

The three JPEG progressive modes are defined as follows (JPEG uses the term “scan” instead of layers, the first two modes are often denoted as sequential progressive modes):

- Spectral selection: the first scan contains the DC coefficients from each block of the image, subsequent scans may consist of a varying number of AC coefficients, always taking an equal number from each block. A typical choice is to encode all DC coefficients into the first scan, subsequently groups of 6 and 7 AC coefficients are organized into one scan.
- Successive approximation: scans are organized according to the binary representation of the coefficients. The first 6 bit of a coefficient is the smallest fraction which the JPEG standard allows to specify. This fraction is coded as in baseline JPEG, while the following bits are emitted without coding. According to the standard DC and AC coefficients have to be treated separately. A typical setting is to use 6 scans (first 6 bit of all DC coefficients Huffman coded, 1 bit more of DC coefficient data, 1 bit more of DC coefficient data, first 6 bit of all AC coefficients Huffman coded, 1 bit more of AC coefficient data, 1 bit more of AC coefficient data).

- Hierarchical progressive mode: an image pyramid is constructed by repeated weighted averaging and downsampling. The lowest resolution approximation is stored as JPEG (i.e. the first scan), reconstructed, bi-linearly upsampled, and the difference to the next resolution level is computed and stored as JPEG file with possibly different quantization strategy (similar to P and B frames in MPEG). This is repeated until the top level of the pyramid is reached.

The JPEG standard also allows to mix different modes. Note that the three modes allow a different amount of scans. Whereas spectral selection offers a maximum of 64 scans, the hierarchical progressive mode is restricted to 5 – 6 sensible scans (given a $2^8 \times 2^8$ pixels image). Successive approximation is restricted to 6 scans (assuming 8 bpp grayscale data). Similar to the scalability profiles of MPEG-2, the JPEG progressive modes are not used very much and are poorly supported and documented in commercial software.

Although providing much better functionality for transmission based applications, the compression performance could be expected to decrease using JPEG progressive modes. This would of course not favour the use of these techniques in transparent encryption scenarios. We have shown in earlier work [15] that provided coding options are chosen carefully, compression performance equivalent to and even exceeding the baseline system may be achieved. All tests concerning the sequential progressive modes were conducted using the IJG's (Independent JPEG Group) reference library, the hierarchical mode is a custom implementation based on the IJG software [15]. All results in this work employ the Lena image with 512^2 pixels and 8bpp.

Figs. 1(a) and 1(b) show how the different scans contribute to the overall file size. This is important knowledge for subsequent transparent encryption since we want to design computationally efficient schemes. In the spectral selection case (Fig. 1(a)) each scan contains one coefficient from each block and as it is expected, the size of the scan decreases for increasing coefficient frequency.

In our example, successive approximation uses the scan configuration used as an example above. We realize that the two scans containing the single DC coefficient bits do not contribute much to the overall file size, whereas the three scans corresponding to single AC coefficient bits contribute 21% and 44% to the overall data.

Table 1 shows two examples for the hierarchical JPEG case using 6 scans (6 pyramid levels), the first optimized for good compression performance (note that in this case the quality of the base layer needs to be low, in our example it is set to $q_f = 10$ [15] resulting in a total of 48589 bytes), the second optimized for a high quality base layer ($q_f = 95$, resulting in a total of 53113 bytes).

It is clearly visible that the distribution of the amount of data among the scans differs significantly depending on the compression settings. This also implies that encrypting e.g. layer 4 only implies a variation in encryption amount between 2.55% and 18.89% of the entire data (which is rather significant since the overall data volume differs only by 10% in our example).

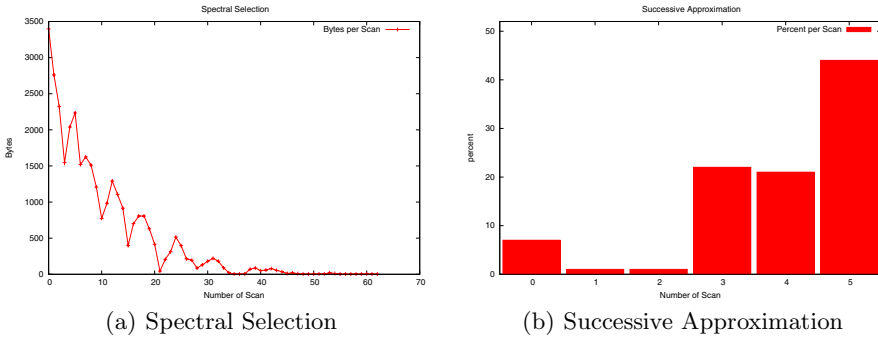


Fig. 1. Data distribution across different scans for sequential progressive modes

Table 1. Percentage of the overall file size contributed by the single layers

Layer	0	1	2	3	4	5
$q_f = 10$	0.6%	0.7%	0.8%	1.3%	2.6%	94.1%
$q_f = 95$	0.7%	1.3%	2.7%	6.9%	19.0%	69.4%

3 Transparent Encryption

3.1 The Classical Approach

The classical approach for transparent encryption of visual data in layered representation is to simply encrypt the enhancement layers, successively encrypting more and more data starting at the end of the file. The remaining scans (i.e. the base layer or scans left in plaintext) may be expected to contain data corresponding to the visual information in lower quality. As we shall see, this approach implies that large amounts of data need to be encrypted to provide a sufficient quality decrease. As an alternative we will investigate strategies where visually more important data, which is not located in the last portions of enhancement information, is encrypted first. The goal is to have similar results as compared to the classical approach but less encryption effort.

Note that in most transparent encryption scenarios the encryption of DC coefficient data has to be avoided since otherwise luminance information is entirely or partially lost and the result is a severely alienated image which might refrain a potential customer from getting interested in the data (quality requirement is not met). This immediately results in a lower bound in achievable image quality that may be achieved with the sequential progressive modes: this bound is attained by reconstructing the image based on DC coefficient data only as shown in Figs. 4 and 6. The situation is more complicated for the hierarchical mode due to the flexibility in its coding parameters. Depending on the quality of the

base layer (and the depth of the pyramid) employed, reconstructions using the image pyramids' base only may vary to a large extent in quality (see Fig. 2 for corresponding examples of a two layer pyramid with high quality $q_f = 95$ and low quality $q_f = 10$ base layers).

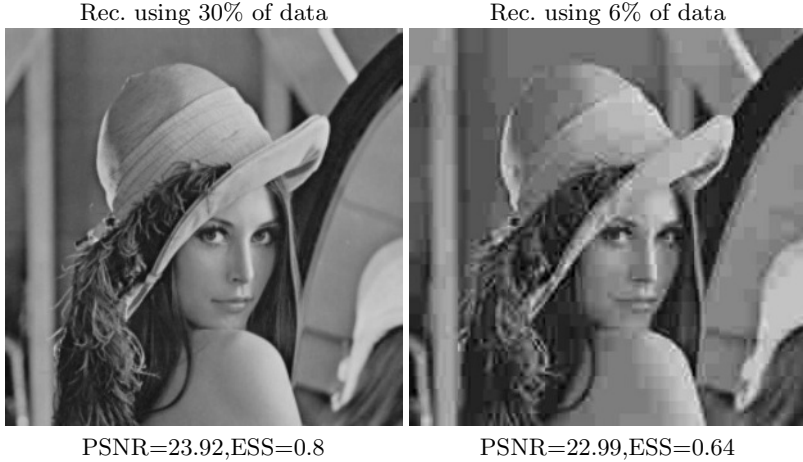


Fig. 2. Image reconstruction based on the base layer only

Decoding a partially encrypted image by treating the encrypted data as being unencrypted leads to images severely degraded by noise type patterns (which originate from the encrypted parts). Using these images to judge the security of the system leads to misinterpretations since a hostile attacker can do much better. In particular, an attacker could simply ignore the encrypted parts (which can be easily identified by statistical means) or replace them by typical non-noisy data. This kind of attack is called “error-concealment” [18] or “replacement attack” [14] in the literature. The IJG software ignores empty scans during decoding – therefore a simple error concealment attack sets the scans affected from encryption simply to zero. In the hierarchical JPEG case we set residual pyramid levels to zero if affected by encryption, the base layer is replaced by uniform gray value 128. See also [7, 15] for these attacks against DCT-based coding/encryption schemes. In order to assess the quality of the visual material after reconstruction in addition to visual inspection we use PSNR and ESS (Edge Similarity Score [12]), the latter measuring the similarity of dominating edges on a block basis in the range $[0, 1]$.

Figs. 3(a) and 3(b) show PSNR values when starting encryption at the end of the file and successively increasing the amount of data encrypted, for spectral selection and successive approximation, respectively, using direct reconstruction and under an error concealment attack. We clearly note the effect of the attack which improves the reconstructions by 4 – 5 dB.

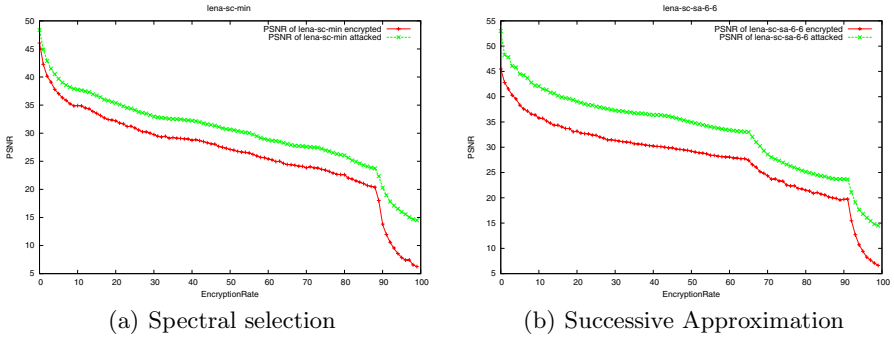


Fig. 3. Increasing the amount of data encrypted

The result curves bend sharply towards lower quality when DC coefficient data is reached for both cases at about 90% of the data encrypted (which again documents that encrypting DC coefficient data violates the quality requirements of transparent encryption), in the successive approximation case we additionally observe different behaviour also when the 6 significant AC coefficient bits are met at about 65% of the data encrypted.

Fig. 4 gives a visual example of the effectiveness of the conducted attack against transparent encryption of 89% spectral selection data. The attack improves the visual quality and PSNR values considerably.



Fig. 4. Transparent encryption of spectral selection (89% encrypted)

This example shows a dilemma which makes the parameters for transparent encryption difficult to adjust. If the amount of encryption is selected to deliver optimal quality without the assumption of a conducted attack (not too good to motivate viewers to pay for better quality and not too low to raise the viewers’

interest in the material), the quality is too high after a successful attack has been mounted. In this scenario, customers able to perform a corresponding attack will probably do so instead of paying. In case the amount of encryption is adjusted to deliver optimal quality assuming an attack has been mounted, the non-attacked material is of rather low quality and might not be suited to raise the average viewers' interest. Therefore, a compromise between those two strategies has to be found. Also, the decision which strategy is applied of course also depends on the business model and the target customer group of the overall application scenario.

3.2 Reducing Encryption Effort

We have seen that the necessity of meeting the security requirement leads to the encryption of a large amount of data in case encryption precedes from the end of the file to the beginning as done traditionally. However, since the last layers do not contribute much to the image quality, it may be more reasonable not to start encrypting at the end of the data, but at a specific point after the DC data according to the required image quality. For most applications starting right after the DC data will be appropriate, in order to minimize the image quality and the encryption rate. Fig. 5 shows an example where encrypting the first AC coefficient only (7% of the file size) results in almost the same image quality as when encrypting 70% of the data starting from the end of the file in the case of spectral selection. We result in a much more efficient transparent encryption scheme employing this idea.

The same observations may be made and similar solution strategies can be applied in case of successive approximation. Using the traditional approach, large quantities of data need to be protected to meet both security and quality requirements, respectively (see for example Fig. 6 where 92% of the data are encrypted: all AC data plus the two single DC coefficient bit scans). Considering the results shown in Fig. 3(b), it is evident that the scan containing the 6 bit AC coefficient data mainly influences the image quality. However, when encrypting this scan we have to process 22% of the overall data accordingly, which is still too much for most applications. One possibility is to split up this scan using spectral selection: Fig. 6 shows results for the encryption of the leading 5 AC coefficients (only the 6 most significant bits, which represent 14% of coefficient data, are encrypted), which leads to similar image quality as the encryption of about 92% of the data with the traditional approach. Again, we were able to significantly reduce the encryption effort as compared to the traditional technique.

Contrasting to the sequential progressive JPEG modes, hierarchical JPEG can offer a great flexibility in its coding parameters. However, as we have seen in the example of Fig. 2, using the traditional approach of encrypting enhancement layers starting at the end of the file at the highest layer) requires the encryption of 70% or 94% of the overall data in the examples of the two layer scenario. When the number of layers is increased, a higher and higher percentage of data has to be encrypted using this technique. Therefore we apply the same principle

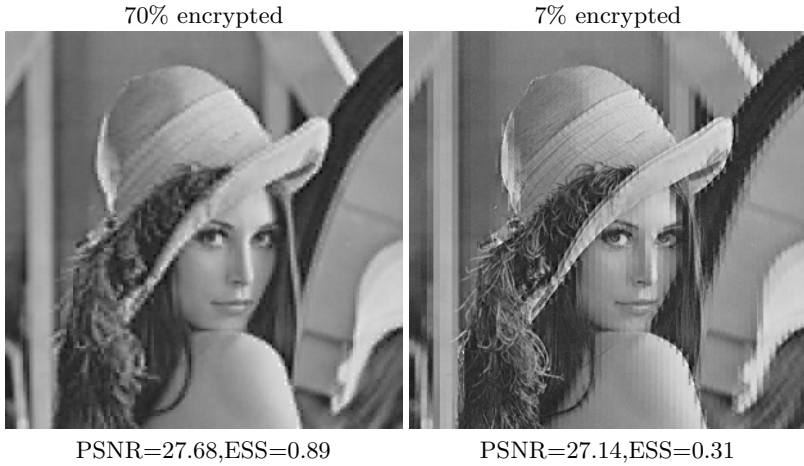


Fig. 5. Efficient transparent encryption of spectral selection (after attack)

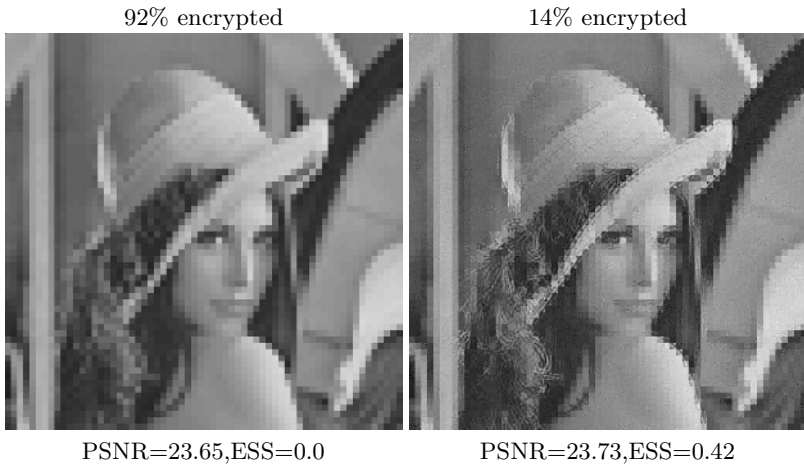


Fig. 6. Efficient transparent encryption of successive approximation (after attack)

as discussed before and encrypt scans between the base layer (layer 0) and the highest enhancement layers. Table 2 shows corresponding results when this idea is applied to the two variants of 6-level pyramids given as an example at the end of section 2 (and we also provide the amount of data in percentages contained in the different layers).

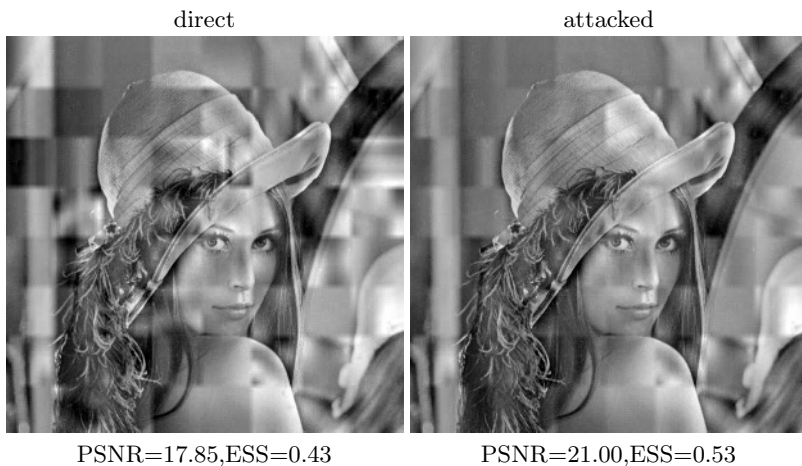
Results in the table indicate that we may reduce the necessary encryption amount significantly using this approach. However, we notice an enormous gap in the quality results between the direct reconstruction and the result obtained by the error concealment attack. Note the extreme example when encryption layer 3 where the difference in PSNR between the directly reconstructed image

Table 2. Results of protecting various layers when applied to the compression optimized pyramid ($q_f = 10$) and to the quality optimized pyramid ($q_f = 95$)

Layers encrypted	0	1	2	3	4	5
$q_f = 10$, % enc.	0.6	0.7	0.8	1.3	2.6	94.1
PSNR direct	12.7	18.4	17.9	11.8	13.8	8.8
PSNR attacked	18.4	19.5	21.0	21.9	22.9	24.6
ESS direct	0.69	0.56	0.43	0.39	0.42	0.35
ESS attacked	0.81	0.61	0.53	0.49	0.57	0.47
$q_f = 95$, % enc.	0.7	1.3	2.7	6.9	19.0	69.4
PSNR direct	16.7	20.5	20.0	14.5	8.5	8.8
PSNR attacked	18.4	22.9	25.1	26.9	26.6	23.9
ESS direct	0.64	0.51	0.40	0.43	0.47	0.53
ESS attacked	0.77	0.66	0.62	0.67	0.68	0.79

and the attacked version is more than 10 dB ! This fact makes it extremely difficult to adjust the encryption parameters properly to meet both, security and quality requirements (which is a difficult task in any case as discussed earlier): the quality of a directly reconstructed image must not be too low, otherwise the average customer will lose interest; but an attacker in this case will succeed in generating a high quality version.

As a consequence, for real life applications we have to rely on settings minimizing this quality gap. Table 2 shows that this gap is by far less pronounced when layer 1 or 2 are encrypted only. Fig. 7 displays the case of encrypting layer 2 for the compression optimized pyramid ($q_f = 10$), Fig. 8 shows the case of encrypting layer 1 for the quality optimized pyramid ($q_f = 95$). If the quality requirements are met for the target application, these settings are a very good

**Fig. 7.** Encryption of layer 2 (0.80% encrypted), compression optimized pyramid

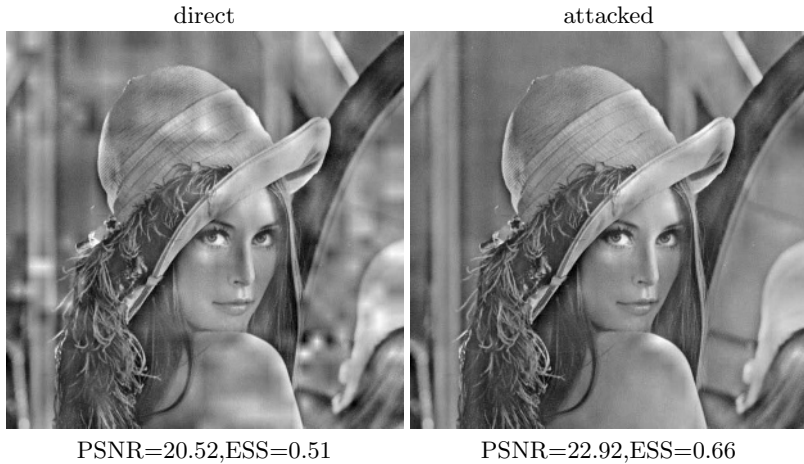


Fig. 8. Encryption of layer 1 (1.25% encrypted), quality optimized pyramid

choice since the encryption effort is very small (0.80% and 1.25% of the overall file size) and the security is rather satisfactory since the discussed gap is rather small in these cases.

Table 2 reveals an additional property when avoiding to encrypt the highest enhancement layer(s): as can be seen, the gap between quality using direct reconstruction and attacked visual data is maximal when the high layers are encrypted. Therefore, besides reducing the encryption amount as suggested in this work we also improve the applicability in real-world scenarios of the scheme by encrypting layers more closely to the base layer.

4 Conclusions

Progressive and hierarchical JPEG may be used for transparent encryption in an efficient manner due to the scalable data format. Parsing the file and searching for the data to be protected can be avoided in this fashion. We have shown that the traditional approach applied to scalable data which starts encryption from the end of the bitstream (enhancement layer encryption) suffers from high encryption demands. The same functionality can be achieved by protecting data situated between base and enhancement layers while reducing the computational encryption effort significantly.

References

- [1] Y. Bodo, N. Laurent, and J.-L. Degelay. A scrambling method based on disturbance of motion vector. In *Proceedings of ACM Multimedia 2002*, pages 89–90, Juan Le Pins, France, December 2003.

- [2] H. Cheng and X. Li. On the application of image decomposition to image compression and encryption. In P. Horster, editor, *Communications and Multimedia Security II, IFIP TC6/TC11 Second Joint Working Conference on Communications and Multimedia Security, CMS '96*, pages 116–127, Essen, Germany, September 1996. Chapman & Hall.
- [3] Jana Dittmann and Ralf Steinmetz. Enabling technology for the trading of MPEG-encoded video. In *Information Security and Privacy: Second Australasian Conference, ACISP '97*, volume 1270, pages 314–324, July 1997.
- [4] Jana Dittmann and Ralf Steinmetz. A technical approach to the transparent encryption of MPEG-2 video. In S. K. Katsikas, editor, *Communications and Multimedia Security, IFIP TC6/TC11 Third Joint Working Conference, CMS '97*, pages 215–226, Athens, Greece, September 1997. Chapman and Hall.
- [5] Marc Van Droogenbroeck. Partial encryption of images for real-time applications. In *Proceedings of the 4th 2004 Benelux Signal Processing Symposium*, pages 11–15, Hilvarenbeek, The Netherlands, April 2004.
- [6] Marc Van Droogenbroeck and Raphaël Benedett. Techniques for a selective encryption of uncompressed and compressed images. In *Proceedings of ACIVS (Advanced Concepts for Intelligent Vision Systems)*, pages 90–97, Ghent University, Belgium, September 2002.
- [7] Mark M. Fisch, Herbert Stögner, and Andreas Uhl. Layered encryption techniques for DCT-coded visual data. In *Proceedings (CD-ROM) of the European Signal Processing Conference, EUSIPCO '04*, Vienna, Austria, September 2004. paper cr1361.
- [8] T. Kunkelmann and U. Horn. Partial video encryption based on scalable coding. In *5th International Workshop on Systems, Signals and Image Processing (IWSSIP'98)*, pages 215–218, Zagreb, Croatia, 1998.
- [9] Thomas Kunkelmann. Applying encryption to video communication. In *Proceedings of the Multimedia and Security Workshop at ACM Multimedia '98*, pages 41–47, Bristol, England, September 1998.
- [10] B. Macq and J.J. Quisquater. Digital images multiresolution encryption. *The Journal of the Interactive Multimedia Association Intellectual Property Project*, 1(1):179–206, January 1994.
- [11] Benoit M. Macq and Jean-Jacques Quisquater. Cryptology for digital TV broadcasting. *Proceedings of the IEEE*, 83(6):944–957, June 1995.
- [12] Y. Mao and M. Wu. Security evaluation for communication-friendly encryption of multimedia. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'04)*, Singapore, October 2004. IEEE Signal Processing Society.
- [13] W.B. Pennebaker and J.L. Mitchell. *JPEG – Still image compression standard*. Van Nostrand Reinhold, New York, 1993.
- [14] M. Podesser, H.-P. Schmidt, and A. Uhl. Selective bitplane encryption for secure transmission of image data in mobile environments. In *CD-ROM Proceedings of the 5th IEEE Nordic Signal Processing Symposium (NORSIG 2002)*, Tromsø-Trondheim, Norway, October 2002. IEEE Norway Section. file cr1037.pdf.
- [15] Thomas Stütz and Andreas Uhl. Image confidentiality using progressive JPEG. In *Proceedings of Fifth International Conference on Information, Communication and Signal Processing, ICICS '05*, pages 1107–1111, Bangkok, Thailand, December 2005.
- [16] A. Uhl and Ch. Obermair. Transparent encryption of JPEG2000 bitstreams. In P. Podhradsky et al., editors, *Proceedings EC-SIP-M 2005 (5th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services)*, pages 322–327, Smolenice, Slovak Republic, 2005.

- [17] A. Uhl and A. Pommer. *Image and Video Encryption. From Digital Rights Management to Secured Personal Communication*, volume 15 of *Advances in Information Security*. Springer-Verlag, 2005.
- [18] Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin. A format-compliant configurable encryption framework for access control of video. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6):545–557, June 2002.
- [19] C. Yuan, B. B. Zhu, M. Su, Y. Wang, S. Li, and Y. Zhong. Layered access control for MPEG-4 FGS. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'03)*, Barcelona, Spain, September 2003.

Preserving TCP Connections Across Host Address Changes

Vassilis Prevelakis¹ and Sotiris Ioannidis²

¹ Computer Science Department, Drexel University, Philadelphia, PA 19104, USA
119613

² Computer Science Department, Stevens Institute of Technology, Hoboken, NJ
07030, USA

Abstract. The predominance of short-lived connections in today's Internet has created the perception that it is perfectly acceptable to change a host's IP address with little regard about established connections. Indeed, the increased mobility offered by laptops with wireless network interfaces, and the aggressive use of short DHCP leases are leading the way towards an environment where IP addresses are transient and last for short time periods. However, there is still a place for long-lived connections (typically lasting hours or even days) for remote login sessions, over the network backups, *etc.* There is, therefore, a real need for a system that allows such connections to survive changes in the IP addresses of the hosts at either end of the connection.

In this paper we present a kernel-based mechanism that recognizes address changes and recovers from them. Furthermore, we discuss the security implications of such a scheme, and show that our system provides an effective defense against both eavesdropping and man-in-the-middle attacks.

1 Introduction

Applications based on the Internet Protocols generally assume that the address of a given node remains the same over long periods of time. Long term connections, especially those that use connection-oriented protocols such as TCP, rely on this assumption to allow connections that may “last months” and can even survive temporary disruptions to the network. The assumption of address immutability is, however, increasingly difficult to sustain. Mobile nodes (*e.g.*, laptops, phones, PDAs, *etc.*) can change addresses as they move from one network to another, but even fixed nodes connected to the Internet via dial-up or DSL link have addresses that change every time their connection is reset. In some cases ISP's initiate such address changes to force users that need a permanent (static) IP address to pay for one. Unfortunately, established connections (*e.g.*, ssh sessions) do not survive the address change, because they rely on fixed source and destination IP addresses. In order to protect these connections when one of the endpoints gets a new IP address, some kind of mechanism is required to allow

both ends of the connection to update the addresses associated with that connection, or to continue using their initial addresses through address translation or tunnels.

In this paper we examine various techniques that have been proposed to address this problem and describe a new technique based on “redirects.” We also discuss the security implications of the use of such mechanisms and propose a novel technique that prevents third parties from hijacking connections.

2 Connection Redirection

To better understand the redirect protocol, consider the following scenario shown in Figure 1 where the communication between two hosts is disrupted because the IP address of one of the hosts changes.

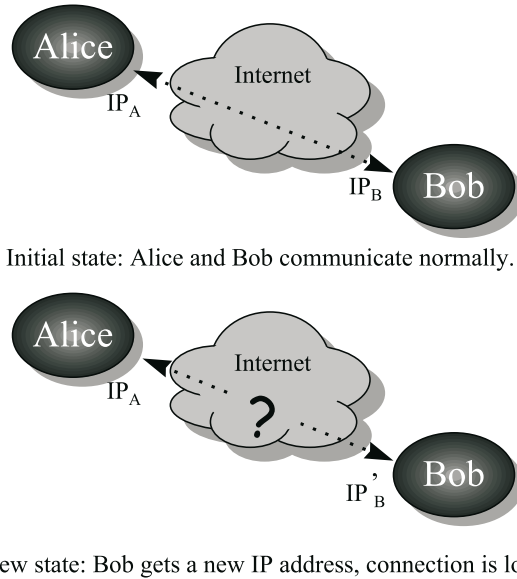


Fig. 1. When Bob acquires a new IP address, all established connections with Alice will be lost

Assume two hosts, Alice and Bob, with IP addresses IP_A and IP_B respectively. Initially, Alice and Bob establish a TCP connection and communicate normally. At some stage in the communication, Bob is forced to acquire a new IP address (IP'_B). At that point if Alice sends a packet to Bob's original address, IP_B , she will either get an ICMP error (*e.g.*, ADDRESS UNREACHABLE), or the packet will be lost (silently). Moreover, if another host has grabbed IP_B , Alice will get a TCP RST. In the first and last cases Alice will immediately

know that the connection has been lost and she will tear down her side of the connection, but in the second case Alice will have to wait until the connection times out.

In every case, the connection will be lost and will have to be reestablished. To retain the connection, both sides must update the state of the IP tables in the network stack, changing all instances of IP_B to IP'_B . Since Bob knows that his address has changed, he can effect the local changes, but he also has to inform Alice via an Address Change Message (ACM, shown in Figure 2), so that she can update her network state.

Old Source IP	New Source IP	Destination IP	Source Port	Destination Port	Authentication	Nonce
------------------	------------------	-------------------	----------------	---------------------	----------------	-------

Fig. 2. Format of the Address Change Message

The ACM may affect all established TCP connections between Bob and Alice causing all applicable network stack entries to be updated. Alternatively, it may apply only to a specific TCP connection. In the latter case Bob will need to send individual ACMs for every TCP connection between himself and Alice.

While the above protocol can be used to redirect TCP connections one has to be sure that this mechanism cannot be used to hijack existing connections. Specifically, the protection mechanism must address both packet injection attacks and data modification attacks.

Packet Injection. A packet injection attack is one where an attacker sends a specially crafted ACM causing an existing connection to be redirected to a host controlled by the attacker. This attack is facilitated by the ability of the attacker to eavesdrop on the communication channel and, hence, is particularly likely in wireless networks where it is trivial to monitor network traffic.

To protect against such attacks, we can either use some pre-arranged secret to guard the ACM, or use sequence numbers that the attacker is unlikely to guess (unless, of course, the attacker is able to monitor the traffic). The extent to which we want to protect hosts from redirect attacks will define which defence mechanism one should use. Understanding the specific types of packet injection attacks will help us form protection guidelines.

The ability to eavesdrop on the communication opens additional avenues of attack such as replay attacks whereby the attacker records an earlier exchange involving an ACM and reuses it to acquire a connection at some point in the future. In order to understand this attack consider the following scenario.

In a LAN where address assignment is handled via DHCP, the attacker can trigger an address change on a host causing it to emit an ACM message. A further address change will cause the victim host to move to a different IP

address allowing the attacker to use the original ACM to redirect traffic back to the freshly released address.

Data Modification. In a man-in-the-middle scenario, the attacker is able to inspect and modify packets exchanged between the two communicating parties. While this attack is harder to carry out in general, if the attacker is in the same LAN as the victim, ARP spoofing can be used to allow traffic to flow through the attacker.

To appreciate the difficulty of countering data modification attacks, consider the use of a shared secret to protect the ACM. If the secret is sent during the current session, the attacker will be able to intercept it and modify it. Thus, the two parties must exchange the secret ahead of time, or use a trusted third party to introduce them to each other.

2.1 Global Versus Local Redirects

When redirecting connections we can opt to send a single redirect request which applies to all active sessions between the two hosts, or send ACMs for each connection separately.

By using individual ACMs for each connection we leverage TCP's sequence numbers in order to ensure that the connection is unlikely to be hijacked because the attacker must guess not only the source and destination ports, but the sequence number as well. If the attacker cannot eavesdrop on the connection, they are unlikely to be able to guess the correct combination or even to mount a brute-force attack by trying all possible combinations due to the size of the search space.

For global redirects we can use a token established at the beginning of the session to authenticate the redirect request. This token can be a 32-bit or 64-bit quantity sent in the original connection request packet. Either side can use that token if it needs to send a redirect packet.

Neither of the two solutions above address the problem of an eavesdropping attacker. Of course, if an attacker is able to monitor the traffic on a connection, he or she will be able to mount a great variety of attacks against that session (including message injection). However, it may not be acceptable to allow the attacker to redirect the entire session.

A good way to address this problem is not to send the token in the clear but to use Diffie Hellman exchange to establish a token, known by both sides but unavailable to a potential eavesdropper. [17] The DH exchange, however introduces extra overhead, is vulnerable to a man in the middle attack and may allow an attacker to mount a denial of service attack on either side by forcing them to perform repeated DH exchanges.

We have addressed these issues with a two level technique that (a) limits the expensive negotiation to long-lived sessions and (b) allows information from previous sessions to be used to enhance the level of security and resist man-in-the-middle attacks as well. The latter technique is similar to the authentication mechanism employed by ssh, that is to transfer the host key when the first

connection between the two hosts is made. Moreover, since the authentication procedure involves the two hosts and not the individual connections we can further reduce overheads by allowing one redirect message to affect all the connections between the two hosts.

3 Design Considerations

We have developed a prototype to test our methods and provide feedback on the efficacy of the various techniques described the previous section. It is important to observe here that we are primarily concerned with authentication and, as we shall see later, data integrity; privacy is not our concern and hence we avoid encryption and its associated overhead. Existing systems such as ssh or TLS may be used as needed to satisfy privacy requirements.

3.1 Initial Key Exchange

As we have seen above, in order to construct a system which is resistant to both data monitoring (eavesdropping) and man-in-the-middle attacks, we need to set up a session key. This is done as part of the initial TCP/IP handshake (Figure 3).

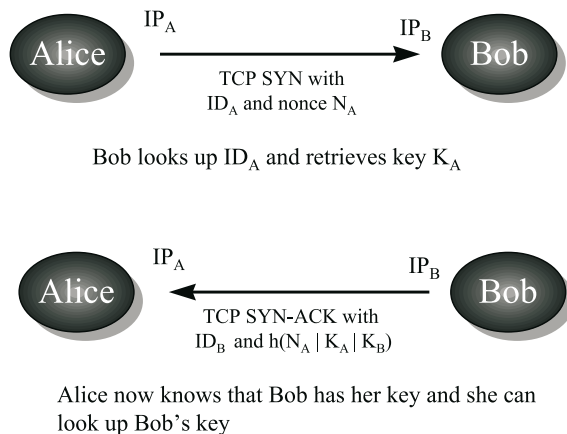


Fig. 3. Session key agreement between parties that know each other. Alice sends her ID along with a nonce to Bob. Bob uses the ID to find Alice’s key and then responds with his ID along with the hashed value of the nonce along with their two keys.

Assuming that Alice and Bob have met before, they already know each other’s secret key. This is used in the negotiation for the session key. Note that Bob does not have a single key that he gives to all his friends, but rather maintains a list in the form:

$ID_A K_A K_B$ timestamp

The reason is that if hosts always use the same key for all their transactions, a malicious host M could contact Alice to get her key and then contact Bob to get his key and thus be in a position to impersonate either one. We prefer this method of authentication as opposed to using public key cryptography because our method is vastly cheaper in terms of CPU requirements.

The timestamp field is used to indicate the last time the host was contacted to allow for pruning of the list to avoid maintaining old and potentially useless information.

3.2 First Contact

If Alice and Bob have never met before, or if either host has purged the key information from its database, they will need to exchange keys. This operation is not performed by the kernel, as it involves a lot of operations that are better done in user-land.

We have modified the network code in the OpenBSD kernel to call an application (keyserv) during the initial stages of the session key exchange (Figure 4). The keyserv program maintains a table with known hosts so that the storage and management of these keys may be managed by the user.

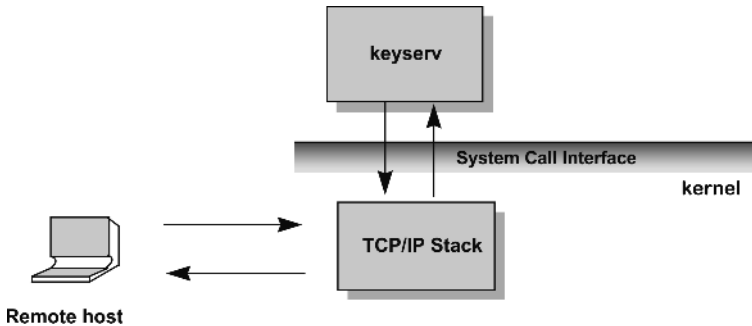


Fig. 4. When an ACM-aware connection is initiated, the kernel asks a user-level application to carry out the session key negotiation

3.3 Short Versus Long Lived Connections

Based on typical TCP/IP connection lifetimes we observe that a great many connections are short lived, and do not require the use of our mechanism. For such connections we should not attempt to use our mechanism to avoid burdening the end-systems with the associated overhead. But how can we determine whether a connection is likely to require the use of the connection redirection system? Initially we based our decision on the service (*i.e.*, we activated our protocol based on the port used for the connection). This allowed us to use the mechanism for, say, ssh connections, but not http. This approach assumes that we have a good understanding of the type of services in use, the use various ports used and that each service can be nicely categorized as short- or long-lived.

So we looked for a more flexible mechanism and we ended up using the following heuristic: *if a connection lasts longer than 10 seconds, it is likely to be a long-lived connection*. We, therefore, arrived at the state model shown in Figure 5.

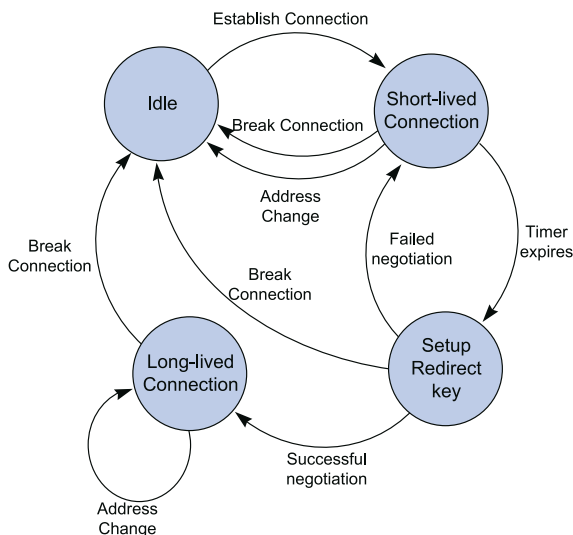


Fig. 5. State diagram showing the activation of the address redirection system

Initially the host is Idle (no connections). When a remote host connects, we start a 10 second timer. Before the timer expires, we treat the connection as a short-lived one, so that if an address change occurs, the connection is dropped. Assuming that the connection is still up when the timer expires, we establish the redirect key and enter the “long-lived” state where our mechanism can be used to recover from address changes. This technique allowed us to reduce overheads and yet has proven to be extremely accurate in predicting the long-lived connections.

3.4 Data Integrity

In RFC-2385 [9] the authors suggest that BGP sessions can be protected through the use of MD5 hashes. The proposed technique involves calculating the hash of the packet to which we have appended a “password.”

In our system we can do something similar using the session key as password and utilizing a more secure algorithm instead of MD5 (see discussion by Dobbertin [7] on the problems with the MD5 hash). This scheme will enhance the typical TCP session with integrity checks (but will not provide privacy) at a very small overhead, since we have already carried out the key establishment negotiation. Since this integrity support is available to long-lived sessions, it is activated after the timer expires (see Figure 5 above).

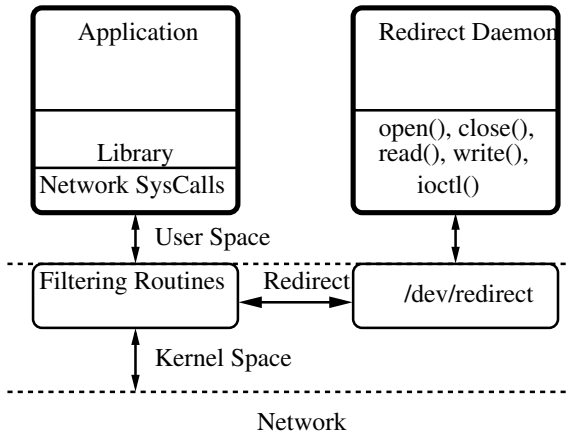


Fig. 6. Block diagram of the redirect implementation. IP addresses are modified on the fly using filtering routines to maintain the network connections.

4 Implementation

We implemented the redirect architecture under OpenBSD 3.6 [1] as a proof of concept. Our implementation consists of three components: (1) a set of kernel extensions, that are responsible for locating and enabling the modification of IP addresses dynamically in the kernel; (2) a user-level daemon process, which implements the redirect monitoring system; and (3) a device driver, which is used to modify the IP data structures according to the requirements of the redirect monitor. Our prototype is very lightweight, consisting of a few hundred lines of C code.

Figure 6 shows a graphical representation of the system, with all its components. The core of the redirect mechanism lives in kernel space and is comprised of the filtering routines and the device driver. The redirect monitoring engine lives in user space, inside the redirect daemon process. Any incoming or outgoing IP packets go through the filter and are subject to possible redirect.

Kernel Extensions. We implemented a set of kernel extensions to permit to modify the Protocol Control Block of existing network connections. This functionality is supported by two operations: `search_pcb` and `modify_pcb`. More specifically, we search all the Protocol Control Blocks for connections of interest and then we modify them accordingly. In the case of a local IP address change, we need to modify the source address of every existing connection. Whereas in the case of a remote IP address change, we only need to modify existing connections to that remote host.

Redirect Device. To maximize the flexibility of our system and allow for easy experimentation, we decided to make the redirect daemon a user level process. To support this architecture, we had to implement a *pseudo device driver*, `/dev/redirect`, that serves as a communication path between the user-space

redirect daemon, and the IP packet redirect engine in the kernel. Our device driver, implemented as a loadable module, supports the usual operations (`open(2)`, `close(2)`, `read(2)`, `write(2)`, and `ioctl(2)`).

```

struct redirect_request {
    int          local_or_remote;
    in_addr_t    oldIPAddress, newIPAddress;
};

for (ever) {
    if (IP local address change) {
        update Protocol Control Block with new source IP address
        for (every existing connection)
            notify remote redirect daemons
    } else if (IP remote address change) {
        update Protocol Control Block with new destination
        IP address
    }
}

```

Fig. 7. Pseudocode of redirect daemon

Redirect Daemon. The last component of our system is the redirect daemon. It is a user-level process responsible for making decisions on whether to redirect IP packets or not. These decisions are based on the changes of the host IP address.

The redirect daemon continuously monitors the network interface for IP address changes. When it detects a change it starts executing the protocol described in Section 3. The redirect daemon of the host that experienced the IP address change, communicates with the redirect daemon of all the network hosts it has established connections to notify them of the IP address change. It then issues a call to the redirect device driver to update the in-kernel Protocol Control Block tables of the existing connections with the new IP address. Similar update actions are taken by the remote redirect daemons upon the receipt of the notification (see Figure 7).

5 Related Work

The problem of maintaining existing connections when the IP address changes is not unique to home networks. As with cell phone networks, Mobile IP (MIP) systems have to address the problem of “handoff” *i.e.*, what happens if the cell phone or mobile PC moves from one area to another [16].

However, MIP systems must also satisfy additional requirements related to the roaming nature of mobile users. In particular, what happens whenever a mobile PC remains disconnected from the network for a significant amount of

time (*e.g.*, during a long flight, or over the weekend) [18]. Another issue is how to ensure that other computers can establish connections to the mobile PC while it moves from network to network.

5.1 Forwarding Node

These problems necessitate the use of a forwarding node that has a fixed IP address [3,5,6]. The mobile PC contacts the forwarding node in order to send and receive packets. A popular way of handling this transparently is via an overlay network: the mobile PC establishes a tunnel with the forwarding node and sends all packets over the tunnel (*i.e.*, the tunnel is designated the default route), while the forwarding node performs NAT on the packets using the tunnel. Other hosts think that packets from the mobile PC originate from the forwarding host due to the use of NAT, while incoming packets are sent over the tunnel to the mobile PC. Packets flow though the tunnel oblivious to the changes of the IP address of the mobile PC.

We have a similar setup in operation for almost 8 years. The forwarding station is an OpenBSD machine connected to the network, while the “mobile PCs” are located in home networks connected via DSL or cable modems. The home networks use an embedded system (running a special version of OpenBSD that boots off a compact flash memory device) acts as an integrated firewall and VPN gateway [15,14].

We use IPsec in tunnel mode to implement the overlay network (Figure 8). Our recent implementation based on OpenBSD 3.0 has been in continuous operation for almost three years (1022 days uptime) demonstrating that sessions can survive even migrations between ISPs (Verizon DSL to Comcast cable).

However, the overlay network technique has two major limitations: one is that it requires a forwarding node with a fixed, globally unique IP address, and the other is that packets always have to take a detour via the forwarding station in order to reach hosts in the Internet. The latter both increases latency and leaves the system vulnerable to failures in the forwarding node, the network hosting the forwarding node, or the transit networks linking the mobile PC to the forwarding node [2].

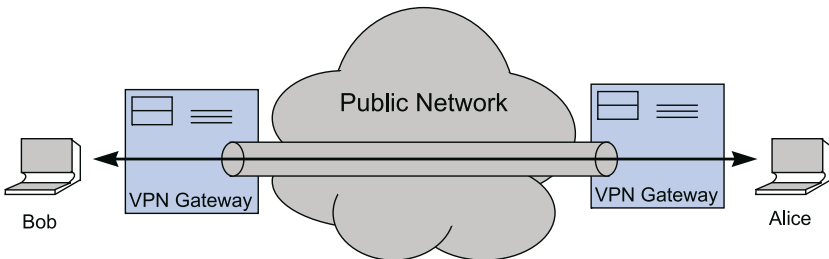


Fig. 8. The overlay network hides changes in the external addresses of the gateway hosts. Internal hosts (Bob and Alice) can use their own addresses all the time.

While these issues may be acceptable in the case of a truly mobile PC, we do not believe that they are acceptable in a home setting, where the end-user is unlikely to be willing to shoulder the cost of the forwarding station or the latency imposed by the detour to the forwarding node.

5.2 Proxy-Based systems

Another way to handle address changes is to use a common rendezvous host [3,19,8]. Both sides connect to the third part that has a fixed address. In this way even if one or even both hosts change addresses, they can locate each other via the common host. This technique is used mainly by voice telephony (VoIP) applications. Such applications also have the added benefit of using connection-less protocols allowing much flexibility in dealing with address changes.

5.3 Mobile IPv6

The design of IPv6 has created a special address class (link-local addresses) within the huge available address space. Link-local addresses are not routable, but are unique and serve to identify a host within a LAN. Though the use of tunneling, hosts can communicate using their link-local addresses and hence be immune to address changes in the intervening network. A special mechanism based on two new destination option fields within packets (binding update and binding acknowledgement) are used to facilitate the updates to the tunnels [13,12,4,10,11].

6 Conclusions

We have presented a system that allows TCP connections to survive addresses changes in the communicating hosts. Our system is designed to allow existing connections to be migrated to new IP addresses without the knowledge or cooperation of the application. Under our system, when an address change occurs, all instances of the original IP address in the kernel IP tables, are dynamically replaced with the new address. Remote systems that have established connections are also notified so that they can update their own data structures.

Recognizing that without adequate safeguards this procedure would create serious security problems, we have implemented a comprehensive security mechanism that protects connections from hijacking even against man-in-the-middle attacks.

Care has been taken to minimize the costs associated with this mechanism, both by reducing the computational overheads and by deferring the expensive cryptographic operations until we are reasonably sure that the connection is in fact long-term and can, therefore, benefit from our services. Another benefit of our approach is that having carried out the necessary mutual authentication between the two hosts, we can use this information to provide integrity checking of the connection with almost negligible overhead. We believe that our system combines efficiency and utility and we would like to see it become a standard feature of all IP-based systems.

Acknowledgements. This work was supported in part by the National Science Foundation under grants DUE-0417085 and CCR-0331584.

References

1. The OpenBSD Operating System. <http://www.openbsd.org/>.
2. N. Aghdaie and Y. Tamir. Client-Transparent Fault-Tolerant Web Service. In *Proceedings of the 20th IEEE International Performance, Computing, and Communications Conference*, April 2001.
3. I. F. Akyidiz. Mobility Management in Current and Future Communications Networks. *IEEE Network*, 12(6):39–49, July/August 1998.
4. P. Bhagwat and C. Perkins. A Mobile Networking System based on Internet Protocol (IP). In *Proceedings of USENIX Symposium on Mobile and Location Independent Computing*, pages 69–82, August 1993.
5. A. T. Campbell, J. Gomez, S. Kim, Z. Turanyi, and C. Y. Wan. Comparison of IP Micromobility Protocols. *IEEE Wireless Communications*, pages 72–82, February 2002.
6. A. T. Campbell, J. Gomez, S. Kim, Z. Turanyi, C. Y. Wan, and A. G. Valko. Design, Implementation and Evaluation of Cellular IP. In *IEEE Personal Communications, Special Issue on IP-based Mobile Telecommunications Networks*, June/July 2000.
7. H. Dobbertin. The Status of MD5 After a Recent Attack. *RSA Labs' CryptoBytes*, 2(2), Summer 1996.
8. D. Funato, K. Yasuda, and H. Tokuda. TCP-R: TCP mobility support for continuous operation. In *IEEE International Conference on Network Protocols*, pages 229–236, October 1997.
9. A. Heffernan. RFC 2385: Protection of BGP Sessions via the TCP MD5 Signature Option. Request for Comments, Internet Engineering Task Force, August 1998.
10. John Ioannidis, Dan Duchamp, and Gerald Q. Maguire Jr. IP-Based Protocols for Mobile Internetworking. In *Proceedings of SIGCOMM*, pages 235–245. ACM, September 1991.
11. John Ioannidis. *Protocols for Mobile Internetworking*. PhD thesis, Columbia University in the City of New York, 1993.
12. D. Jonhson and C. Perkins. Mobility Support in IPv6. Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.
13. C. Perkins. RFC 2002: IP Mobility Support. Request for Comments, Internet Engineering Task Force, October 1996.
14. Vassilis Prevelakis and Angelos Keromytis. Designing an Embedded Firewall/VPN Gateway. In *Proceedings of the International Network Conference*, 2002.
15. Vassilis Prevelakis and Angelos Keromytis. Drop-in Security for Distributed and Portable Computing Elements. *Journal of Internet Research*, 13(2), 2003.
16. P. Stuckman. *The GSM Evolution*. Wiley, 2003.
17. Gong Su. *MOVE: Mobility with Persistent Network Connections*. PhD thesis, Columbia University, New York, New York, 2004.
18. R. Zhang, T. F. Abdelzaher, and J. A. Stankovic. Efficient TCP Connection Failover in Web Server Clusters. In *Proceedings of IEEE InfoCom*, March 2004.
19. S. Zhuang, K. Lai, I. Stoica, R. Katz, and S. Shenker. Host Mobility using an Internet Indirection Infrastructure. In *First International Conference on Mobile Systems, Applications, and Services (ACM/USENIX Mobisys)*, May 2003.

A Security Architecture for Protecting LAN Interactions

André Zúquete¹ and Hugo Marques²

¹ IEETA / University of Aveiro, Portugal

² DEE / ESTCB, Portugal

Abstract. This paper describes a security architecture for a LAN. The architecture uses the 802.1X access control mechanisms and is supported by a Key Distribution Centre built upon an 802.1X Authentication Server. The KDC is used, together with a new host identification policy and modified DHCP servers, to provide proper resource allocation and message authentication in DHCP transactions. Finally, the KDC is used to authenticate ARP transactions and to distribute session keys to pairs of LAN hosts, allowing them to set up other peer-to-peer secure interactions using such session keys. The new, authenticated DHCP and ARP protocols are fully backward compatible with the original protocols; all security-related data is appended to standard protocol messages.

1 Introduction

In the last decade a major effort has been made, and is still being made, to provide basic security mechanisms to communication networks. However, we still lack a basic security architecture for dealing with security issues in Local Area Networks (LANs). A LAN is not usually considered a dangerous computing environment; instead it is regarded as a trusted network environment. Therefore, all management and configuration activities occurring between hosts in a LAN are not protected at all. The result is that the management of LANs is highly sensitive to several local attacks using eavesdropping and origin spoofing techniques.

Spoofing attacks launched locally in a LAN usually require a direct connection of the attacker to the LAN. In other words, require an inside attacker. Inside attackers can be prevented to a certain level by physical barriers, ensuring that only trustworthy people are connected to the same LAN. But computers in a LAN can be compromised by cyberplagues, therefore trusting only on people for building up the confidence in the correct exploitation of a LAN is a dangerous assumption to take.

Furthermore, LANs are not shared only by trustworthy people, or even by people that know each other. For instance, cabled LANs used as distribution networks for Wireless LANs (WLANs) deployed in hot spots are used by communities of users that only have in common an authorization for accessing the network. Such LANs cannot be regarded at all as trusted network environments by the people using them.

1.1 Contribution

In this paper we propose a security architecture for cabled LANs capable of providing a flexible set of security attributes for local communications within the LAN. Each and every LAN host can choose a set of mandatory security attributes for interacting with other local hosts. We will call to such hosts Secure LAN hosts (or SLAN hosts for short). Each SLAN host can decide on its particular protection policy and SLAN peers should negotiate pairwise agreements ensuring the most secure set of common security requirements.

The SLAN architecture does not impose a single protection model for all hosts. Instead, it provides a basic service for local, authenticated secret key distribution. This service is then used to build a flexible set of secure communication mechanisms within the LAN. Namely, each host uses the keys distributed to enforce a set of security requirements towards other SLAN hosts. SLAN hosts can even use differentiated policies for dealing with different LAN interactions.

The SLAN architecture provides protection mechanisms for LANs independently of the hardware used for connecting LAN hosts. Namely, it allows hosts to get local privacy even when using hubs and origin authentication even when not using switching devices with port filtering capabilities. This fact simplifies the deployment of secure LANs without requiring sophisticated hardware; all that is required from the network hardware is a basic support of the 802.1X authentication framework [1], in order to authenticate SLAN hosts or users.

In the design of the SLAN architecture we tried to make it as transparent as possible for the users of LAN hosts. Consequently, we designed a Key Distribution Centre (KDC) extending the 802.1X authentication and authorization service [1], which is being deployed in many WLANs but can also be used in cabled LAN scenarios. Keys distributed by the KDC are provided to hosts in specific, critical interactions, namely DHCP and ARP configurations [2,3].

A prototype was implemented on Linux hosts, for testing the KDC and the new, secure ARP. The prototype also uses session keys distributed along secure ARPs to set up IPsec Security Associations between pairs of hosts. But, for lack of space, the prototype will not be described.

To avoid confusions, in this paper we will use the term MAC as an acronym of Message Authentication Code and L2 address instead of MAC (Medium Access Control) address. We will also use the expression $[D] \& \text{MAC}(K)$ to denote the set formed by the datum D and its MAC computed with key K .

The paper is structured as follows. Section 2 overviews security threats in cabled LANs. Section 3 presents the SLAN architecture. Section 4 evaluates the security of SLAN. Section 5 describes related work and compares it with SLAN. Finally, Section 6 presents the conclusions and the future work.

2 Security Threats in Cabled LANs

LANs provide an illusion of security because users typically disregard the probability of inside attacks. In this section we will provide a few examples on how insecure LANs can be, in order to motivate our work.

Layer 1 Insecurity. This type of security breach is related with physical access to the networking infrastructure. If not protected, attackers can get access to it and, in most cases, perform eavesdropping and compromise network availability and performance. The difficulty of tapping depends on the technology used: it's straightforward in WLANs, easy in cabled LANs using copper cables, and hard, or impossible, in cabled LANs with fiber optics.

Because users need to have access to network resources, layer 1 security based on physical protection is very difficult to implement. Network administrators usually put network infrastructure devices inside closed racks but this only protects those devices from direct manipulation. Network plugs are still distributed along working areas, so anyone can plug in and access the network.

In this scenario, it seems that the use of IEEE 802.1X specification could be the solution [1] to limit the access to the network infrastructure only to authorized users. Even if a user gets access to the medium through a network plug, he must still provide valid credentials to have access to the remaining network. But once a user gets connected, he gets access to all the information that “passes by” his network plug. If the network has a broadcast nature, then information generated by one user can be “viewed” by all other users. Typically, network administrators resolve this issue using layer 2 or layer 3 switching devices. This solution, however, has some flaws as described below.

Layer 2 Insecurity. Layer 2 security issues are mainly related with addressing and switching. ARP [3] is the only protocol responsible for providing layer 3-to-layer 2 address mappings. Since no message authentication is provided, ARP is vulnerable to spoofing, modifying and replaying attacks. Through the manipulation of ARP *Request* and *Reply* packets, it's possible to corrupt ARP cache tables of remote devices; this attack is called *ARP cache poisoning*. Using this vulnerability, an attacker can execute Denial-of-Service (DoS) and Man-in-the-Middle (MitM) attacks. GrabitAll¹ is a publicly available tool that can perform such attacks.

Several solutions for this problem have been devised, namely:

- Static ARP entries in the ARP cache table, disabling ARP transactions. This approach is only suitable on small, very stable networks, but these are exactly the ones where ARP spoofing is not a major concern.
- *Port security* features on switching devices, preventing the L2 address associated with a port to change once set. This approach requires sophisticated hardware support and management policies for releasing mappings between access ports and L2 addresses.
- Detection techniques alerting when different L2 addresses are detected for the same layer 3 address. But since ARP *Replies* are unicast messages, these tools fail to detect ARP poisoning initiatives in switched networks.
- Cryptographically protected ARP transactions. Crypto-Ethernet NIC [4], SLL [5], S-ARP [6], Secure ARP [7] and TARP [8] are cryptographic solutions that will be discussed in Section 5.

¹ <http://ntsecurity.nu/toolbox/grabitall>

The other layer 2 security issue is switching. Network administrators typically replace hubs with switches to restrict frames forwarding only to the required network segments, and not all. This means that other network segments will not see this traffic, hence eavesdropping is impossible. However, switches can be misled by attackers sending frames with spoofed L2 addresses, allowing them to get frames targeted to the spoofed host. In this scenario an attacker can execute a DoS or a MitM attack against a specific victim. GrabitAll, a publicly available tool previously referred, can perform such attacks.

Another way of eavesdropping on a switched network is by “converting” a switch on a hub by interfering with cache tables of source L2 addresses. Etherflood² is one of many publicly available tools that perform this kind of attack. In environments with multiple switches, eavesdropping or DoS attacks can also be launched by using spanning-tree protocol manipulation [9].

Both addressing and switching vulnerabilities can be minimized through the use of Virtual LANs (VLANs). VLANs permit a physical network to be divided into a set of smaller logical networks, making communication between LANs only possible through the use of a router. Because the attack techniques described before are only possible in layer 1 and 2 switching environments, by using VLANs network administrators reduce the scope of attacks. VLANs, however, suffer the “VLAN hopping” security issue [9].

Layer 3 Insecurity. One major security issue of layer 3 concerns routing. Because routers, by default, don’t authenticate routing updates, an attacker can send bogus routing packets and impersonate himself as a router, and perform DoS or MitM attacks. Of course, one can use static routing instead of dynamic, but that just doesn’t scale well. So the solution resides on configuring authentication on routing protocols. The problem with this is that not all routing protocols support authentication, for example RIPv1 [10] and IGRP.

Dynamic routing on a LAN, using ICMP *Redirect* datagrams [11], does not also provide any form of authentication. Attackers can configure default gateways of local LAN hosts using false ICMP *Redirect packets*, spoofing both IP and L2 addresses of the real gateway. The redirection can serve different purposes, such as DoS (redirection into a black hole) or MitM attacks. Note that this is a very convenient way to get access to all the traffic sent by a victim on a switched LAN. Hosts can be configured to reject all ICMP *Redirect* packets, but that reduces the flexibility in the management of LAN routes.

The most accepted solution to mitigate layer 3 vulnerabilities is IPSec [12]. IPSec can provide authentication, integrity, confidentiality and anti-replay services, both for IPv4 and IPv6, in two different modes: transport and tunnel. However, IPSec is not easy to manage and its key distribution protocol — IKE [13] — requires some sort of secrets bound to hosts. This fact complicates the deployment of IPSec protection upon a user authentication protocol.

² <http://ntsecurity.nu/toolbox/etherflood>

3 SLAN Architecture

The previous section showed that protecting the traffic on a LAN is much more than just protecting a single communication layer or a single protocol. In fact, we need to protect each and every interaction capable of being spoofed or tampered by a local attacker. Thus, protecting interactions in a LAN requires an integrated security architecture, rather than a set of independent, non-cooperative security solutions, one for each protocol used in a LAN. The security architecture must provide a minimum set of services capable of authenticating users accessing the LAN and authenticating basic host configuration activities. Finally, the security architecture should also provide the basic means for protecting, in many different, configurable ways, all the remaining interactions within a LAN.

Looking at the usual behavior of hosts in LAN, they follow the next sequence of steps:

- Get connected to the LAN and eventually authenticated. Authentication is usually skipped in cabled networks, being more frequent in wireless networks.
- Run a DHCP transaction in order to get a set of network configuration parameters. The most relevant parameters are the IP address of the DHCP client and the network mask; other important parameters are the IP addresses of the default gateway and of the local DNS server.
- When using IP for the network protocol, run ARP transactions for getting proper mappings between IP and L2 addresses within the LAN.
- Communicate using the parameters obtained with DHCP and the IP-L2 mappings in ARP cache tables.

Thus, for protecting a LAN we need to protect each of these separate actions. This means that the SLAN architecture should (i) enforce the authentication of hosts (users) that get connected to the LAN (ii) enforce the correct configuration of hosts using DHCP, ensuring that they get the proper configuration parameters to operate in the LAN, (iii) enforce the correct population of hosts' ARP cache tables and (iv) provide hosts with session keys and mechanisms for protecting other local, peer-to-peer (P2P) interactions.

3.1 Overview

The services provided by the SLAN security architecture are the following:

1. Protected DHCP. This implies the authentication of all replies provided by a genuine DHCP server and the detection of replays of past replies.
2. Protected ARP. This implies the authentication of ARP *Request* and *Replies*, namely the authentication of the IP-L2 mappings provided, and the detection of past request/replies.
3. Protection of other network protocols between any pair of hosts in the LAN. In this case we don't know anything about the protocols that are going to be used. Therefore, what as to be done is to provide peers with key material suitable for enforcing the origin of the data and/or its confidentiality using ad-hoc security mechanisms.

All these services are supported by secret, shared session keys between interacting hosts. Session keys are associated with pairs of SLAN hosts. For the first service, the DHCP client must share a session key with a genuine DHCP server; the key should allow both DHCP client and server to authenticate data in DHCP messages. The key should also allow them to detect and reject replays of old, authenticated DHCP messages. For the second service, the ARP requester must share a session key with the ARP responder; the key should allow the ARP requester to validate the data in the ARP *Reply* and, most important, the legality of the IP-L2 mapping provided. The key should also allow the ARP requester to detect and reject replays of old, authenticated ARP *Replies*. For the third service, each host must share a session key with the peer it is communicating to; the key should allow both peers to protect packets exchanged between them, providing origin and data authentication and/or data confidentiality. The key should also allow receivers to detect and reject replays of previous packets.

For distributing all these session keys, the SLAN architecture has a local KDC. The KDC provides session keys to authenticated hosts willing to interact with other specific hosts. Hosts authenticate themselves against the KDC using a shared, short-term secret key. For distributing session keys among pairs of SLAN hosts we chose to integrate it with the existing DHCP and ARP protocols messages. However, the SLAN versions of DHCP and ARP are fully backward compatible: they use the same standard protocol messages, but these are extended to include extra security-related data. Hosts not SLAN-enabled do not use this extra data but can still get the usual data from the standard DHCP and ARP message fields. SLAN-enabled hosts, on the contrary, should require secure DHCP and ARP interactions, but can still interact with non-SLAN enabled hosts if allowed by security policies.

3.2 Key Distribution Centre (KDC)

The KDC must share a secret with each SLAN-enabled local host. A possible solution was to force hosts and KDC to share a password, like in Kerberos [14]. However, we chose a more flexible approach, building up the KDC on top of an 802.1X authentication framework [1].

The 802.1X authentication framework was designed to provide authentication, authorization and key distribution services for cabled or wireless LANs. In the 802.1X terminology, hosts are called *Supplicants*. When a Supplicant first accesses the LAN, it gets connected to an *uncontrolled port*, which lets it interact only with an *Authentication Server* (AS) through an *Authenticator*. After a successful EAP-based authentication protocol [15] between the Supplicant and the AS, both the Supplicant and the Authenticator end up with a shared, secret key generated and distributed by the AS — the *Pairwise Master Key*, PMK. When the Supplicant gets the PMK, it runs the four-way handshake with the Authenticator and, finally, gets connected to a *controlled port*, with full access to the network resources allowed by the Authenticator.

In the SLAN architecture, the KDC is build upon an 802.1X AS (see Fig. 1). The SLAN client first runs an 802.1X authentication protocol with the AS/KDC

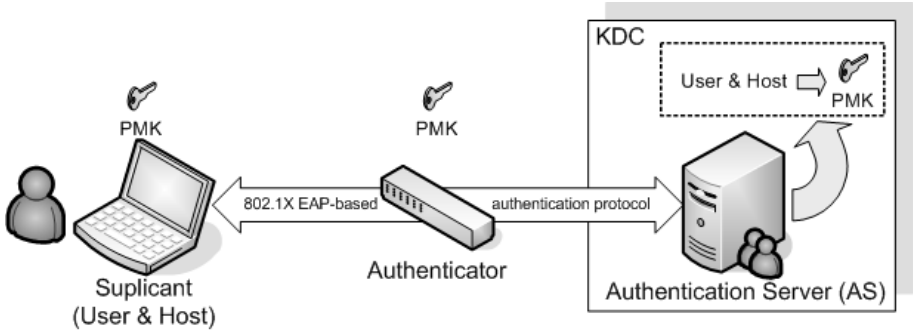


Fig. 1. Relationship between the 802.1X authentication framework and the SLAN KDC. The KDC is build upon the AS and stores mappings between resulting PMK keys and the corresponding users and hosts. The triplets $\langle username, PMK, NID \rangle$ are computed both by the KDC and the authenticated Supplicant.

and both end up with a PMK after a successful authentication of the former; the four way-handshake with the Authenticator is irrelevant for our architecture. A PMK will then be the authentication secret shared between the KDC and an authenticated SLAN host, and gets associated to some identification of the user — username, X.500 Distinguished Name, e-mail address, etc. — that ran the 802.1X authentication protocol (see next section).

SLAN administrators are free to choose the best EAP-based authentication protocols for 802.1X authentication; that is completely transparent for the SLAN key distribution. However, the SLAN trust model requires confidence on the KDC, therefore all EAP-based authentication protocols ran within the 802.1X must require mutual authentication, to properly authenticate the AS/KDC.

3.3 Network Identification

Before describing how key distribution happens, we will first describe how hosts are identified within a SLAN. This is a critical issue for distributing the right keys to the right hosts and users.

As we already saw, L2 addresses can be manipulated at will. This means that it is possible to have two hosts with equal L2 addresses in the same LAN without being able to decide which one is legitimate, if any. Therefore, we need to have some mechanism for distinguishing SLAN hosts using equal L2 addresses. Simultaneously, this mechanism should preserve network privacy, thus usernames are not acceptable for identifying users in SLAN-related packets.

We decided to use a dynamic, collision free numerical identification tag for authenticated hosts. This tag, hereafter referred by **Network ID (NID)**, is the digest of two values: the username and the PMK of the (authenticated) user that is using the host:

$$NID_{host} = \text{digest}(username, PMK_{username, host})$$

	Peers	Main message contents	MAC key
1	$h_1 \rightarrow \text{KDC}$	$\text{NID}_1, \text{NID}_2, \text{Options}$	PMK ₁
2	$h_1 \leftarrow \text{KDC}$	$\{\text{SK}_{1,2}\}_{\text{PMK}_1}, \text{KeyProp}_{1 \rightarrow 2}, \text{NID}_2 \text{ info}$	
3	$h_2 \leftarrow h_1$	$\text{another message}, \text{KeyProp}_{1 \rightarrow 2}$	SK _{1,2}

$$\text{KeyProp}_{1 \rightarrow 2} \equiv \{\text{NID}_1, \text{NID}_2, \text{SK}_{1,2}, \text{Options}\}_{\text{PMK}_2}$$

Fig. 2. Basic protocol for distributing a session key SK_{1,2}, provided by the KDC, from host h_1 to host h_2 ; PMK₁ and PMK₂ are the secret keys shared between the KDC and the users using hosts h_1 and h_2 , respectively; NID₁ and NID₂ represent the NID of hosts h_1 and h_2 , respectively

The term *username* means some form of user identification used in the 802.1X authentication process. The KDC gets $\langle \text{username}, \text{PMK} \rangle$ pairs from the 802.1X AS, computes a NID from them and keeps a table of $\langle \text{username}, \text{PMK}, \text{NID} \rangle$ triplets. Authenticated hosts can compute by themselves the NID after getting the PMK (see Fig. 1).

Hereafter, we will simply use the expression NID_{*i*} for referring the NID of the host h_i and the expression PMK_{*i*} for referring the PMK of the user using host h_i . Naturally, there is a direct correspondence between NID_{*i*} and PMK_{*i*} values for each authenticated user in the SLAN.

3.4 KDC Key Distribution

Session keys distributed by the KDC are associated to pairs of NID values: a NID identifies a session key requester and the peer that will share that key. However, since on a SLAN we need to manage correct mappings between IP and L2 addresses, key distribution will also consider such mapping.

The KDC key distribution paradigm closely follows Kerberos: an authenticated host sends a *Key Request* message for requiring a session key to interact with other (authenticated) host; and the KDC replies with a *Key Reply* message, containing a new, random session key encrypted with the PMK of both peers (messages 1 and 2 in Fig. 2). The data block containing the session key and encrypted with the peer PMK is similar to a Kerberos ticket; we will call it a **key propagator**. The requester is responsible for sending the received key propagator to the peer. Such interaction will not happen isolated, but instead by piggybacking key propagators in other messages (see message 3 in Fig. 2 and messages 4 in Figs. 3 and 4).

Besides a session key, the key propagator includes the NID of the peers associated to the enclosed session key and an *Options* field. This field is intended to disseminate, in an authenticated way, relevant characteristics of the host requesting the key propagator. These characteristics may a priori be known by KDC or may be proposed by the requester. Examples of possible uses are (i) the IP and L2 addresses or (ii) relevant administration capabilities (e.g., DHCP server, IP gateway, etc.) of the requester. The value of the *Options* also depends on the value of the *Options* field in the KDC *Key Request*.

The session key requester may also get, from the KDC, some relevant information about the other host that is going to share the session key. This information is provided in the **NID *info*** field of the KDC reply, authenticated by the message MAC.

Unlike Kerberos tickets, our key propagators do not have timestamps nor a lifetime. The reason for not having them is that it is highly unlikely to someone be able to reuse a key propagator some other time, because it depends on a short-term, temporary key — the PMK currently held by the target host — while, in Kerberos, tickets are always encrypted using long-term keys, shared between Kerberos and principals.

A KDC is free to choose and use a caching policy for the provided session keys. Each time a host requests a new session key for interacting with another host, the KDC generates a new session key or provides a previous one, if still in the cache. This cache is not critical for the system to work, but it can improve its efficiency and resilience to DoS attacks.

Hosts are free to cache session keys, indexed by peers' NID, for the time they may consider more appropriate, and may require new session keys at any time. But the lifetime of session keys is closely tied to the lifetime of PMK keys, since the renewal of a host PMK implies the renewal of the corresponding NID. This means that a cached session key becomes automatically stale whenever the corresponding PMK expires.

3.5 Authenticated DHCP

The first step of a host that gets connect to a LAN, after an 802.1X authentication process, is to get properly configured for using the LAN — get its IP address and local network mask, the default gateway, the local DNS domain suffix and server, etc. Such parameters are usually provided by DHCP servers.

As stated before, we want DHCP messages to be authenticated for SLAN hosts. Therefore, we need to negotiate session keys between DHCP servers and authenticated DHCP clients. Furthermore, SLAN DHCP clients need to know if they are interacting with the correct DHCP server, and not just with some bogus DHCP server. One possibility was to integrate DHCP services in the KDC, which could facilitate the use of the hosts' PMKs to authenticate DHCP interactions. However, we considered this a very limitative approach, so we designed an authenticated DHCP where KDC and DHCP can easily be implemented by different hosts. The only requirement is that the KDC knows which hosts are DHCP servers and shares a PMK with each of them.

The authenticated DHCP is presented in Fig. 3 and runs like this:

1. The client broadcasts a DHCP *Discover* with its NID.
2. A server uses its NID and the client NID to send a *Key Request* to the KDC.
3. The KDC returns the client username, a session key and a key propagator for the DHCP client.
4. The server uses the username for getting an IP address and sends to the client an authenticated DHCP *Offer* with the key propagator.

	Peers	Main message contents	MAC key
1	$h \rightarrow DS$	DHCP <i>Discover</i> , NID_h	—
2	$DS \rightarrow KDC$	NID_{DS}, NID_h	
3	$DS \leftarrow KDC$	$\{SK_{DS,h}\}PMK_{DS}, KeyProp_{DS \rightarrow h}(Options = DHCP),$ $NID_h \text{ info} = \text{username}$	PMK_{DS}
4	$h \leftarrow DS$	DHCP <i>Offer</i> , $KeyProp_{DS \rightarrow h}(Options = DHCP)$	
5	$h \rightarrow DS$	DHCP <i>Request</i> , $[IP_h, L2_h] \& MAC(PMK_h)$	$SK_{DS,h}$
6	$h \leftarrow DS$	DHCP <i>Acknowledge</i> , $IPL2Prop_h$	

$$IPL2Prop_h \equiv [[IP_h, L2_h] \& MAC(PMK_h)] \& MAC(PMK_{DS})$$

Fig. 3. Authenticated DHCP. All DHCP messages, except the *Discover*, are authenticated with a MAC computed with a session key ($SK_{DS,h}$) requested by the DHCP server (DS) and propagated to the client in the DHCP *Offer*. The *Options* field in the key propagator states that it was requested by a DHCP server. $IPL2Prop_h$, IP-L2 propagator, is an authenticated IP-L2 address mapping build together by the DHCP client and server; IP_h and $L2_h$ represent the IP and L2 addresses of the DHCP client, respectively.

5. The client sends an authenticated DHCP *Request* with an extra, authenticated IP-L2 address mapping.
6. The server sends an authenticated DHCP *Acknowledge* with $IPL2Prop$ — **IP-L2 propagator**, a secure token with the client’s IP-L2 address mapping.

The messages in the authenticated DHCP extend the usual protocol messages, being backward compatible. The extensions can be implemented in two different ways: using the DHCP authentication option [16] or appending the extra data to layer 2 packets.

This protocol uses a radically different approach for identifying DHCP clients. In fact, typical DHCP negotiations deal with the configuration of nodes given their L2 addresses. However, this simple approach is not advised, because attackers could get control of IP addresses pre-allocated for specific L2 addresses belonging to specific hosts. Thus, DHCP resource allocation must evolve and the *Discover* message should include some sort of mapping to usernames to manage pre-allocated IPs, instead of using simply L2 addresses. We used a NID for referring to a username — the same that is used to compute it and is stored by the KDC. Given the client NID, its username is provided to the DHCP server in the $NID_h \text{ info}$ field of the KDC reply. For network privacy, usernames may be transmitted to DHCP servers encrypted with their PMK.

The session key for authenticating the DHCP transaction is requested by the DHCP server, because the DHCP client does not know at the beginning the NID of the DHCP server. The key propagator is sent to the DHCP client in first occasion, i.e., within the DHCP *Offer*. The *Options* field in the key propagator stands that it was requested by an authorized DHCP server. This way, the client can be sure that the offer was produced by a correct DHCP server, and not just by any other authenticated host. All messages, excluding the DHCP *Discover*, are authenticated using a MAC computed with the session key.

The last two messages of the DHCP protocol include, besides the MAC for authentication, two extra values that will be used as an IP-L2 mapping commitment. The client appends an extra version of the mapping, authenticated with its PMK, to the DHCP *Request*; the server further authenticates it with its PMK. The final value, returned to the client, is the IP-L2 propagator.

IP-L2 propagators can be fully validated only by the KDC, allowing it to easily learn current mappings between IP and L2 addresses. This will be useful for authenticating ARP *Requests* and *Replies* (see next section).

3.6 Authenticated ARP

The ARP is the base mechanism to learn mappings between IP and L2 addresses in a LAN. Hosts keep an ARP cache table with such mappings, which are frequently refreshed to ensure accuracy. ARP caches can also be feed with information gathered from other packets.

One of our goals was to assure the correctness of ARP cache tables. Therefore, ARP caches are populated exclusively with (i) information provided by authenticated ARP *Replies* or (ii) information gathered for building ARP *Replies*. Furthermore, during an authenticated ARP run, both hosts should agree on a protection policy concerning their own P2P interaction. Namely, they should say to each other if they want to receive authenticated messages and/or if their interaction should be encrypted. The keys used for securing the P2P interaction should be derived from the session keys distributed along ARP protocol runs.

Therefore, in our SLAN architecture, ARP caches keep five values, instead of pairs of IP and L2 addresses. The quintets are formed by: the peer NID, IP and L2 addresses, the shared session key and the protection policy. The session key is used to protect, according to the protection policy, the communication between the local host and the peer referred by the cache entry.

The authenticated ARP is presented in Fig. 4 and runs like this:

1. h_1 broadcasts an ARP *Request* with its NID, IP-L2 propagator and intended P2P protection policy.
2. The correct responder h_2 uses its NID and the requester's NID to get a new session key from the KDC. We assume that SLAN hosts use a well-known L2 address or a well-known IP-L2 mapping for the KDC. Both values can be provided to SLAN hosts in many ways, namely within extensions of the EAP authentication protocol or by DHCP servers.
3. The KDC returns a session key, a key propagator and the (validated) IP-L2 address mapping of the ARP requester.
4. The responder sends an authenticated ARP *Reply* with the key propagator — providing the P2P session key and assuring the correct IP-L2 mapping of the responder — and the final P2P protection policy.

The *Options* field of the session key request contains the IP-L2 propagators held by h_2 and h_1 . If both are valid, then the KDC will include the IP-L2 address pair of h_2 in the *Options* field of the provided key propagator and the IP-L2 address pair of h_1 in the *NID₁ info* field of its reply.

Peers	Main message contents	MAC key
1 $h_1 \rightarrow h_2$	ARP <i>Request</i> (looking for h_2), $NID_1, IPL2Prop_1, P2P\ Options$	—
2 $h_2 \rightarrow KDC$	$NID_2, NID_1, Options = \langle IPL2Prop_2, IPL2Prop_1 \rangle,$	PMK ₂
3 $h_2 \leftarrow KDC$	$\{SK_{2,1}\}PMK_2, KeyProp_{2 \rightarrow 1}(Options = \langle IP_2, L2_2 \rangle),$ $NID_1\ info = \langle IP_1, L2_1 \rangle$	
4 $h_1 \leftarrow h_2$	ARP <i>Reply</i> , $KeyProp_{2 \rightarrow 1}(Options = \langle IP_2, L2_2 \rangle), P2P\ Options$	SK _{2,1}

Fig. 4. Authenticated ARP. The IP-L2 mapping in the ARP *Request* is validated by the KDC. The ARP *Reply* is authenticated with a session key (SK_{2,1}) requested by the responder and included in the key propagator. The *P2P Options* field allows peers to exchange security requirements for protecting their future interaction, using SK_{2,1} after the execution of the ARP.

At the end of this protocol both h_1 and h_2 have complete and trustworthy quintets in SLAN ARP caches:

- h_1 : $NID_2, IP_2, L2_2$ and $SK_{2,1}$ are provided and authenticated by the key propagator; the P2P protection policy is authenticated by the MAC.
- h_2 : the correct mapping between $NID_1, IP_1, L2_1$ and $SK_{2,1}$ is provided by the KDC reply; the P2P protection policy is the one sent in the ARP *Reply*.

If the KDC keeps a cache of past session keys provided to pairs of hosts, then both hosts may get the same session key on future authenticated ARP transactions, independently of the requester. But if they get a different key, all they have to do is to adjust accordingly the P2P protection mechanisms already being used between them, if any.

3.7 Authenticated and Encrypted Interactions Within a LAN

After an ARP protocol run, both interacting hosts on the SLAN share a session key. This session key can then be used to add security attributes to the traffic between both hosts. Namely, the session key can be used to encrypt data and to compute MACs for integrity control. But such traffic protection can be enforced in some different ways.

One possibility is to apply security mechanisms at the link layer for protecting each packet. Another possibility is to use those session keys to setup a pair of IPSec Security Associations (SAs) between both hosts, one for each communication direction. This is a natural approach, because the session key is obtained after an ARP, which means that both hosts will interact using IP. The IPSec SAs can be set up in two different ways:

1. After setting up a pair of IKE SAs using the IKE protocol and the session key as an IKE pre-shared secret. This allows the hosts to use IPSec security policies for protecting IP traffic without bothering with the management and the configuration of IKE authentication.
2. Directly and off-line by both peers, using only the session key and the set of security policies agreed on the ARP protocol run. This approach does not require any configuration of IPSec policies but may require a richer dialog at the ARP level to support many different protection policies.

Using IPsec has the advantage of using a mature technology for protecting IP traffic, while using the novel key distribution protocol to deal with the authenticated set up of SAs. On the other hand, IPsec is a complex and heavy protocol, and in some operational scenarios it may be too overwhelming to deal with the security risks on a LAN.

3.8 SLAN Novel Security Policies

The SLAN security architecture requires new security policies that break with several traditions in common LAN environments.

The first security policy is that hosts are not free to choose their IP address; they should get it from a DHCP server. Otherwise, they cannot get a proper IP-L2 propagator required for running an authenticated ARP transaction. But this loss of freedom is a natural consequence: for certifying IP-L2 address mappings, the SLAN architecture cannot allow free, ad-hoc mappings defined only by hosts. Server hosts belonging to a SLAN, such as DHCP servers, must run some sort of bootstrap code to get a PMK shared with the KDC.

The second security policy is that DHCP resource reservation moves from L2 address-based into username-based, which is a new identification paradigm within DHCP.

Finally, the third security policy is that the lifetime of PMKs stored in the KDC defines the lifetime of all local credentials — key propagators and IP-L2 propagators — as well as the access to the network through the 802.1X Authenticator. This is very important for simplifying network management, but need to be properly integrated with DHCP. Namely, the lifetime of DHCP leases should be coordinated with the lifetime of hosts' PMKs stored in the KDC and in the 802.1X Authenticator.

4 Security Evaluation

We will now evaluate the security of SLAN taking in consideration the threats presented in Section 2.

L2 Address Stealing. In a SLAN, L2 address stealing no longer helps attackers to get other people's resources from DHCP servers, because these should implement an alternative, username-based resource allocation. However, resource allocation in network switches is still an issue. In fact, switch's cache tables may implement learning mechanisms restricting an L2 address to a single port. In this case, an attacker, possibly an authenticated host, can jam network switches and implement DoS attacks on specific hosts by using a copy of their L2 address.

DHCP Impersonation. SLAN DHCP servers must authenticate their *Offer* and *Acknowledge* replies using a session key that is shared with the DHCP client. This key is provided by a key propagator which can only be generated by the KDC, and includes a flag stating that the peer is an authorized DHCP server.

Key propagators are sent to clients inside authenticated messages, such as a DHCP *Offer*, and their authentication is assured by a MAC computed with the key hidden inside the key propagator. Since the key propagator is encrypted with the client's PMK, an attacker would have to know this key to forge a key propagator and the DHCP *Offer* message where it is used. Therefore, though attackers can forge dummy key propagators, they cannot make good use of them to impersonate DHCP servers in sending a DHCP *Offer* message.

After getting a valid DHCP *Offer*, the client has a session key shared with the DHCP server, allowing them to authenticate the subsequent *Request* and *Acknowledge* messages. Since attackers cannot guess this key, they cannot usefully forge this final part of the DHCP protocol.

Concerning message replies, they can be detected by using the XID field of DHCP messages [2] and the authentication of them all but the *Discover* message.

ARP Poisoning Attacks. SLAN clients build their ARP tables solely from IP-L2 propagators, and not from the ordinary ARP reply fields. Thus, ARP poisoning attacks can only be conducted by using forged or replayed IP-L2 propagators. In the following paragraphs we will show that they cannot be either forged or replayed with useful results.

IP-L2 propagators are jointly build by SLAN hosts and authorized DHCP servers, thus cannot be solely created by hosts. Furthermore, their authenticated information is only useful (i.e., accepted by the KDC) if correctly authenticated by both a host and a DHCP server. Therefore, DHCP servers fully control the correctness of IP-L2 mappings in new IP-L2 propagators.

Authorized DHCP servers cannot be impersonated by attackers because they share a PMK with the KDC, which is used to authenticate IP-L2 propagators. Thus, only authorized DHCP servers can produce valid IP-L2 propagators.

IP-L2 propagators can be captured by attackers but cannot be used by them in a useful way. The initial component provided by a host is authenticated with its PMK, thus only attackers with an equal PMK could make good use of a captured IP-L2 propagator. With long, random PMKs generated in each network authentication, it is highly unlikely to get two equal PMKs, thus preventing the reuse of captured IP-L2 propagators.

5 Related Work

Some solutions have been proposed to tackle layer 2 security problems. The scope of the solution, for the great majority of proposals, resides in protecting only the ARP protocol. In this section we will provide a brief comparison of SLAN with the solutions that we considered more relevant.

The S-ARP [6] approach to authenticate DHCP transactions is somewhat similar to the SLAN approach. However, SLAN uses secret, shared keys and MACs, while S-ARP uses asymmetric keys, certificates and digital signatures, which is a slower approach. Furthermore, S-LAN does not address the problem of stolen L2 addresses. Thus, SLAN provides a more efficient and effective approach for dealing with the secure allocation of network resources using DHCP.

The TARP [8] approach, using tickets to distribute IP-L2 mappings, looks similar to the SLAN approach, though being totally different. Tickets' signed mappings are valid for a specific time, and can not be revoked before, while SLAN mappings, provided by IP-L2 propagators, are valid while PMKs are in use. Therefore, SLAN provides a better solution for preventing the illegal use of such mappings when no longer valid.

Both Crypto-Ethernet NIC and SLL [4,5] distribute session keys between LAN hosts, like SLAN. But they use asymmetric cryptography, while SLAN uses symmetric cryptography. Thus, SLAN should perform better, even when considering the deployment of asymmetric algorithms in NICs' hardware.

Secure ARP [7] uses, like SLAN, a central service with secret keys shared with all the hosts in the LAN. However, Secure ARP uses a totally new service for this, for managing long-term shared keys, while SLAN extends transparently the 802.1X authentication paradigm for using short-term PMKs. Secure ARP does not distribute session keys, while SLAN does. Finally, Secure ARP requires clock synchronization.

None of the solutions referred above prevents unauthorized access to the LAN, and consequently the injection of malicious messages. Furthermore, they keep the usual DHCP policies for allocating resources, i.e., based on the L2 address presented by clients. Also, their goal is to protect specific network transactions, such as ARPs, and not to inter-operate with other protocols. Finally, they require an initial phase where an administrator manually configures keys in each of the LAN hosts.

With the SLAN architecture we follow a different approach, building up from the 802.1X authentication architecture. Only authenticated and authorized hosts access the network, authenticated hosts can get trustworthy DHCP configurations and IP-L2 mappings, and ARP transactions distribute trustworthy IP-L2 mappings and P2P session keys that can be used by other protocol layers, such as IPsec. The use of a novel DHCP policy for allocating IP-L2 mappings makes the network more secure against DoS attacks using stolen L2 addresses. Finally, we don't require synchronized clocks.

6 Conclusions and Future Work

In this paper we presented a new security architecture for LAN environments, either wired or wireless. The architecture is build on top of an 802.1X authentication framework. Namely, a KDC is added to an 802.1X AS, allowing PMKs distributed to authenticated hosts to be used for authentication and key distribution in the LAN. Thus, the management overhead for configuring SLAN hosts is minimal if starting from an 802.1X authentication scenario.

Authenticated hosts are identified by NIDs, instead of L2 or IP addresses, and NIDs are computed from usernames and PMKs. Modified DHCP servers, implementing novel resource allocation policies and authenticated DHCP interactions, allow a correct configuration of SLAN hosts. An authenticated ARP allows SLAN hosts to get trustworthy IP-L2 mappings and to distribute session

keys among LAN peers. Those session keys can then be used by other network protocols to implement other P2P security mechanisms.

A prototype implementation was developed for Linux systems, though not presented here for lack of space. It demonstrated experimentally the suitability of authenticated ARPs and subsequent IPSec configurations based on the negotiated session keys.

There are many research topics for further improving SLAN. First, the definition of novel DHCP resource allocation policies for SLANs and their integration with the lifetime of PMKs in the KDC. Second, the negotiation of P2P security policies within ARPs. Third, the implementation of P2P security mechanisms, namely at layer 2, instead of using IPSec. Forth, the study of several caching strategies for session keys in the KDC and in SLAN hosts. And fifth, the design of security mechanisms and policies for multicast/broadcast traffic on a SLAN.

References

1. IEEE: IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control. IEEE Std 802.1X-2001 (2001)
2. Droms, R.: Dynamic Host Configuration Protocol. RFC 2131, IETF (1997)
3. Plummer, D.: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. RFC 826, IETF (1982)
4. Khoussainov, R., Patel, A.: LAN security: problems and solutions for Ethernet networks. *Computer Standards & Interfaces* (22) (2000) 191–202
5. Hunleth, F.: Secure Link Layer (2001)
http://www.hunleth.com/fhunleth/projects/sll/sll_report.pdf.
6. Bruschi, D., Ornaghi, A., Rosti, E.: S-ARP: a Secure Address Resolution Protocol. In: 19th Annual Computer Security Applications Conf. (ACSAC '03), Las Vegas, NV, USA (2003)
7. Gouda, M.G., Huang, C.: A Secure Address Resolution Protocol. *Computer Networks* 41 (1) (2003)
8. Lootah, W., Enck, W., McDaniel, P.: TARP: Ticket-based Address Resolution Protocol. In: 21st Annual Computer Security Applications Conf. (ACSAC '05), Tucson, AZ, USA (2005)
9. Dubrawsky, I.: SAFE Layer 2 Security In-depth Version 2. White Paper (2004)
http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/sfblu_wp.pdf.
10. Hedrick, C.: Routing Information Protocol. RFC 1058, IETF (1988)
11. Postel, J.: Internet Control Message Protocol. RFC 792, IETF (1981)
12. Thayer, R., Doraswamy, N., Glenn, R.: IP Security Document Roadmap. RFC 2411, IETF (1998)
13. Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). RFC 2409, IETF (1998)
14. Kohl, J., Neuman, C.: The Kerberos Network Authentication Service (V5). RFC 1510 (1993)
15. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748, IETF (2004)
16. Droms, R., Arbaugh, W., Eds.: Authentication for DHCP Messages. RFC 3118, IETF (2001)

Simulation of Internet DDoS Attacks and Defense

Igor Kotenko and Alexander Ulanov

St. Petersburg Institute for Informatics and Automation (SPIIRAS),
39, 14 Liniya, St.-Petersburg, 199178, Russia
{ivkote, ulanov}@iiias.spb.su

Abstract. The paper considers the software simulation tool DDoSSim which has been developed for comprehensive investigation of Internet DDoS attacks and defense mechanisms. This tool can be characterized by three main peculiarities: agent-oriented approach to simulation, packet-based imitation of network security processes, and open library of different DDoS attacks and defense mechanisms. DDoSSim allows deeply investigating various attacks and defense methods and generating valuable recommendations on choosing the best defense. In the paper the agent-oriented approach suggested is considered. The taxonomy of input and output parameters for simulation is outlined. The main DDoSSim components are specified. One of the experiments on protection against DDoS attacks demonstrates some DDoSSim possibilities. We consider different phases of defense operations – learning, decision making and protection, including adaptation to the actions of malefactors.

Keywords: Security modeling and architecture, Security models for ambient intelligence environments, Infrastructure security, Security simulation, DDoS.

1 Introduction

The present theoretical investigations in information security of large-scale systems do not allow security experts to formalize adequately the antagonistic counteraction of network attacks and defense. Though the researchers can represent particular defense mechanisms, the understanding of security systems as holistic entities is a very difficult task. This understanding depends on many dynamical interactions between particular security processes and cyber-counteraction between antagonistic elements. It is especially right, taking into account the evolution of the Internet into decentralized distributed environment where a huge number of cooperating and antagonistic software agents exist and interact.

One of the very dangerous classes of malefactors' attacks is DDoS [16]. Distributed, dynamical and cooperative character of such attacks complicates attack detection and protection. Realizing effective DDoS defense system is a very complicated problem. Effective defense includes the mechanisms of attack prevention, attack detection, tracing the attack source and attack counteraction. Adequate protection can only be achieved by cooperation of different distributed components [17].

The main task of defense systems against DDoS is to accurately detect these attacks, quickly respond to them [26] and recognize the legitimate traffic that shares the attack signature and deliver it reliably to the victim [17]. *Traditional defense* include detection and reaction mechanisms. Different network characteristics are used for detection of malicious actions (for example, source IP address [21], traffic volume [8], and packet content [19], etc.). To detect abnormal network characteristics, many methods can be applied (for instance, statistical [12], cumulative sum, pattern matching, etc). As a rule, the reaction mechanisms include filtering [20], congestion control [14] and traceback [11]. But, as a result of several reasons (detection of DDoS attack is most accurate close to the victim, separation of legitimate is most successful close to the sources, etc.), adequate victim protection to constrain attack traffic can only be achieved by *cooperation of different distributed components* [16, 17]. There are a lot of architectures for distributed cooperative defense mechanisms [2, 4, 10, 17, 19, 26, 27, etc.]. For example, [2] proposes a model for an Active Security System, comprising components that actively cooperate in order to effectively react to a wide range of attacks. COSSACK [19] forms a multicast group of defense nodes which are deployed at source and victim networks. The SOS [10] uses a combination of secure overlay tunneling, routing via consistent hashing, and filtering. A collaborative DDoS defense system proposed in [27] consists of routers which act as gateways. The distributed defense system described in [26] protects web applications from DDoS attacks. The DefCOM system [17] uses a peer-to-peer network of cooperative defense nodes.

In our opinion, it is possible to answer soundly on the questions about defense against network attacks, including DDoS attacks, by modeling and simulation of present and new attacks and defense mechanisms. It is very important to use adequate modeling and simulation approach and powerful simulation environment which give a researcher an opportunity to comprehensively investigate various modes of attack and defense operation, insert new methods, analyze efficiency of defense (for example, false positives, false negatives; percent of normal traffic filtration), etc.

Our research goal is to suggest a common approach and simulation environment for investigation and elaboration of adequate defense methods against DDoS attacks which can produce well-grounded recommendations on the choice of defense mechanisms that are the most efficient in particular conditions. The rest of the paper is structured as follows. *Section 2* outlines the common approach for simulation. *Section 3* describes attack and defense mechanisms used. *Section 4* presents the taxonomy of input and output parameters for simulation. *Section 5* considers the software environment developed and analyses the issues of network topology selection. *Section 7* demonstrates the example of experiments provided. *Conclusion* outlines the main results and future work guidelines.

2 Simulation Approach

We try to use the *agent-oriented approach* to simulate security processes in the Internet. It supposes that the cybernetic counteraction is represented as the interaction of different teams of software agents [6, 24, 25]. The aggregated system behavior becomes apparent by means of the local interactions of particular agents in dynamic

environment that is defined by the model of computer network. We distinguish at least two agent teams: the team of agents-malefactors and the defense team. The agents from the same team collaborate to realize the threat or to defend the network.

It is assumed the competing agents gather information from different sources, operate with uncertain knowledge, forecast the intentions and actions of opponent, estimate the possible risks, try to deceive each other, and react on opponent's actions. The choice of behavior for each team depends on the chosen goal of functioning and is defined dynamically depending on the opposite team actions and the environment state.

The *mechanisms of agent coordination* are based on the three groups of procedures [24, 25]: acts consistency maintenance; agents' functionality monitoring and recovery; and communication selectivity support (to choose the most "useful" communication acts). The models of agent functioning are to foresee, what each agent knows, what task has to be solved and to which agent it must address its request to receive such information if it is outside of its competence. The messages of one agent are to be represented in such terms that are understandable by other agents.

It is supposed that agents are to be able to realize the mechanisms of self-adaptation. The team of agents-malefactors evolves with the aid of generation of new instances and types of attacks and attack scenarios to overcome the defense subsystem. The team of defense agents adapts to the actions of malefactors by changing the security policy, forming new instances of defense and security profiles.

The *conceptual model of agents' counteraction* includes: (1) Ontology of application domain containing application notions and relations between them; (2) protocols of teamwork (for team of malefactors and team of defense); (3) Models of individual, group and team behavior of agents; (4) Communication component for agent message exchange; (5) Models of environment – the computer network, including topological and functional components.

It is proposed to use *various models* to research the processes of cybernetic counteraction. The choice of specific models depends on the necessary simulation fidelity and scalability. For example, analytical models let imitate the global processes happening in Internet, but describe the processes only on an abstract level. Packet-level simulation gives the opportunities to imitate the proceeding processes with high fidelity. They represent the network attack and defense actions as the exchange of packets. The greatest fidelity is archived with the hardware testbeds, but it succeeds in simulating the sufficiently limited fragments of agents' interactions. The approach used in the paper is based on packet-level simulation with the use of tools for network processes imitation as basic level of simulation environment.

3 Attacks and Defense Mechanisms

DDoS attacks agents are divided into two classes: "daemon" and "master". Daemons are attack executors. Master coordinates them. On the preliminary stage daemons and master are deployed on available (already compromised) hosts. The important parameters are the quantity and "distribution" of agents. Then the phase of team establishing takes place. Daemons send to master the messages with information that they

are alive and ready to work. Master stores the information about team members and their status. The malefactor sets the mutual team goal – to start the DDoS attack in the given moment of time. Master receives the attack parameters. Its goal is to send it to all available daemons. Then daemons begin to act. Their local goal is to execute the master instruction. They start to send the attack packets to the given host in the given mode. Master examines daemons periodically to know that they are workable. Master controls the given attack mode by receiving the replies from daemons. When a daemon does not answer, master decides to change attack parameters. For example, it can send the commands to change the attack intensity to all or particular daemons. Daemons can execute the attack in several modes. This influences on the possibility of defense team to detect and block the attack and to trace and defeat the attack agents. The mode can be specified, for example, by the intensity of packet sending (packets per second) or (and) the method of IP address spoofing. The malefactor can stop the attack giving to master the command “stop the attack”. Master resends this command to daemons, and they stop the attack.

Defense agents are classified into the following classes: initial information processing (“sensor”); secondary information processing (“sampler”); attack detection (“detector”); filtering (“filter”); investigation (“investigator”).

In the initial moment the defense agents are deployed on the hosts corresponding to their roles: sensor and sampler – on the way of traffic to the defended host; detector – on any host of defended host subnet; filter – in the entrance to the defended host subnet; investigator – on any host outside of defended host subnet.

Sensor processes information of network packets and collects statistical traffic data for the defended host. Sensor can calculate the amount of traffic (bits per second – BPS) and determine the addresses of hosts that make the largest traffic. The functions of sensor can be fulfilled by *sampler*. Sampler processes the network packets and creates the model of normal functioning for the given network (in learning mode). Then in normal mode it analyses and compares the traffic with the model of normal traffic. It picks out the addresses of hosts that do not correspond to the model and sends them to detector. The examples of methods which can be realized by sampler are Hop counts Filtering (HCF) [9], Source IP address monitoring (SIPM) [22], Bit per Second (BPS), etc.

The *detector* local goal is to make a decision about the beginning of attack on the basis of sensor or (and) sampler data. Detector sends the list of attack addresses received from sensor or (and) sampler to filter and investigator. The *filter* local goal is to filter the traffic on the basis of detector data. The *investigator* goal is to trace and defeat the attack agents. After receiving a message from detector it examines the obtained IP addresses for the presence of attack agents and tries to defeat them.

4 Taxonomy of Input and Output Parameters for Simulation

We differentiate the input parameters which specify DDoS attack and defense mechanisms for simulation.

The scheme of *DDoS attack parameters* is based on the attack taxonomy suggested in [15]. The following criteria were selected:

- *Victim type.* Application, host or network can be chosen. It is necessary to set victim IP address and port.
- *Attack type.* Brute-force (UDP/ICMP flood, smurf/fraggle, etc.) or semantic (TCP SYN, incorrect packets, hard requests).
- *Impact on the victim.* One can choose a disruptive attack (when all daemons attack simultaneously) or a degrading attack (when daemons join the attack one by one). It is easier to detect the attack in the first case.
- *Attack rate dynamics.* It can be constant or variable when the intensity changes in time. The function of changing attack packet rate is given to daemons. The change can be increasing (daemons send more and more packets) or fluctuating.
- *Agents' set permanency.* The set of agents can be persistent (all daemons participate in attack) or variable. In last case master can divide all daemons to several groups and each of them attacks alternately.
- *Possibility of exposure.* The attack can be discovered when it is possible to distinguish the attack packets. We distinguish non-filterable and filterable attacks. In non-filterable attack, the attack packets are formed to be indistinguishable from legitimate. In filterable attack, the attack packets can be discovered by field values, size, exploited protocol, etc.
- *Source addresses validity.* Attacker can use the valid (real) or spoofed source address sending the attack packets. This address can be routable or non-routable. The method of spoofing may be as follows: (1) Without spoofing ("no") – the real address of host (where daemon is deployed) is used; (2) "Constant" – an address is randomly chosen, then it is used for sending the attack packets; (3) "Random" – with every new attack packet a new address from the given range of addresses is randomly chosen. This range does not intersect with the range of addresses used in the given network; (4) "Random real" – with every new attack packet a new address from the given range of addresses is randomly chosen. This range is in the range of addresses used in the given network.
- *Degree of automation.* Attack can proceed automatically after setting the parameters or by the malefactor control. In such a case he (she) can interfere and change one of parameters on all phases of attack. The communication mechanisms between daemons and master can be direct (master knows the addresses of all daemons) or indirect (agents communicate via a server).

The scheme of *DDoS defense parameters* is built on the basis of classification proposed by authors. The criteria selected are as follows:

- *Deployment location:* source, intermediate or defended subnets.
- *Mechanism of cooperation.* The mechanism of particular components operation can be centralized or decentralized. In the last case the defense components are autonomous and can combine their efforts.
- *Covered defense stages.* The stages (mechanisms) the defense method can implement are as follows: (1) attack prevention; (2) attack detection; (3) attack source detection; (4) attack counteraction.
- *Attack detection technique.* There are two types of detection: misuse and anomaly. One chooses one particular detection method or the set of methods.

- *Attack source detection technique.* Attack source detection (or “traceback”) can be realized by packet signatures, packet marking, generation of auxiliary packets, etc.
- *Attack prevention/counteraction technique.* One can use filtering (of packets or flows), resource management (differentiation, change of quantity, roaming) and authentication.
- *Technique for model data gathering.* Data can be generated by learning or be obtained from external sources.
- *Determination of deviation from model data.* One can use thresholds, rules (for packets and connections), determining fluctuation in probabilistic traffic parameters, and data mining (depending on the kind of defense mechanism).

The *output parameters* used to estimate the defense mechanisms are as follows: List of detectable attacks; Time of attack detection (from the start of attack); Time of attack reaction (time from detection to counteraction); Percent of false positives; Percent of false negatives; Percent of normal traffic filtration; Computational complexity (quantity of computational resources used), etc.

5 Simulation Environment

The simulation environment DDoSSim architecture consists of the following components (figure 1): OMNeT++ Framework, INET Framework, Multi-agent & DDoS Framework.

Multi-agent simulation is implemented in Multi-agent Framework that uses the library of DDoS attack and defense mechanisms called DoS Framework. INET Framework is used to simulate the IP nodes. It is an OMNeT++ model itself.

OMNeT++ Framework [18] is a discrete event simulator. Simulation models are composed of hierarchically nested modules that interact due to message passing (figure 1, OMNeT++ Framework: simulation model and component library). INET Framework and Multi-agent DDoS Framework are the OMNeT++ models. The exchange of messages between modules happens due to channels (modules are connected with them by the gates) or directly by gates. A gate can be incoming or outgoing to receive or to send messages accordingly. Channel has the following properties: propagation delay, bit error rate and transmission data rate.

OMNeT++ INET Framework is the OMNeT++ modular simulation suite with a realistic simulation of the Internet nodes and protocols. The highest IP simulation abstraction level is the network itself, consists of IP nodes. IP node can represent router or host. IP node in INET Framework corresponds to the computer representation of Internet Protocol (figure 1, INET Framework). The modules of IP node are organized in such a way like operating system process IP datagram. The module that is responsible for network layer (implementing IP processing) and the “network interface” modules are mandatory. Additionally one can plug the modules that implement higher layer protocols: transport (UDP, TCP, including TCP Sockets; routing: MPLS, LDP, RSVP, OSPF-TE) and application (HTTP, Telnet).

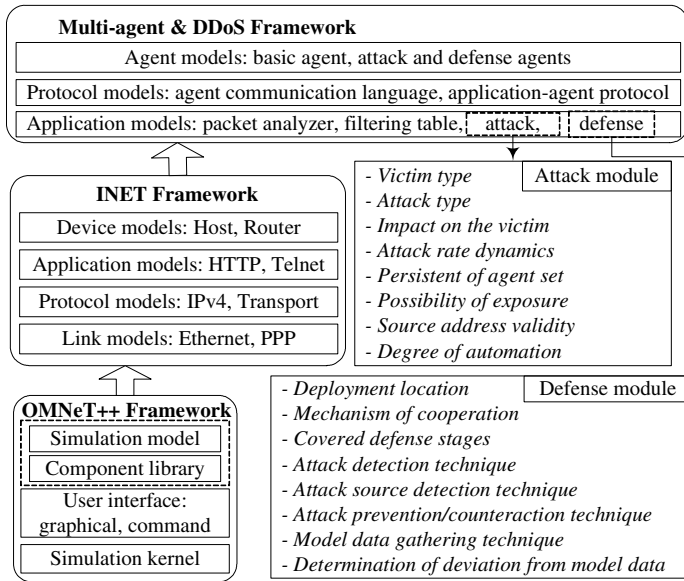


Fig. 1. DDoSSim Simulation environment architecture

Multi-agent & DDoS Framework is the INET Framework modular suite aimed to simulate the DDoS attack and defense mechanisms on the basis of agent team counteraction (figure 1, Multi-agent DDoS Framework). One can distinguish between DDoS Framework and Agent Framework architecturally.

DDoS Framework suite consists of DDoS attack and defense modules (figure 1, Attack module, Defense module) and the modules that expand IP node from INET: the filtering table and the packet analyzer. Attack and defense modules are the applications and are deployed on the network layer of IP node. There were implemented different DDoS attacks and defense mechanisms, for example, Hop-count Filtering (HCF), Source IP address monitoring (SIPM), BPS, etc. To set the DDoS attack conditions it is necessary to define the corresponding parameters, including victim type (host), attack rate dynamics (function of attack packets sending rate), spoofing technique (no spoofing, random, subnet), etc. Also one need to set up the defense parameters, including deployment location (defended, intermediate, source subnet), detection technique, model data gathering technique and its parameters (time interval and time shift of data collection), etc.

Agent Framework consists of modules representing agents which are implemented as applications. There were used the elements of abstract FIPA architecture [7] during agent modules design and implementation. Agent communication language is implemented for agent interactions. The message passing happens above TCP protocol (transport layer). Agent directory is mandatory only for agents that coordinate other agents in teams. Agent can control other modules (including DDoS Framework modules) due to messages.

Agents are deployed on hosts in the simulation environment. Their installation is fulfilled by connecting to the modules serving transport and network layers of protocol

stack simulated in OMNeT++ INET Framework. The *generalized representation of agent* “sampler” structure is depicted in figure 2. Sampler contains the transport layer (depicted as a message), needed to communicate with other agents, network layer (depicted as a blue cube) to collect traffic data and agent kernel (depicted as a shape of human image). The agent kernel contains communication language, knowledge base and message handlers from the neighbor modules. The representation of sampler *deployment into simulation environment* is depicted in figure 3. One can see that the agent is plugged into host through the “tcp” module implementing TCP protocol. Agent is also connected with the “sniffer” module used to analyze network packets.

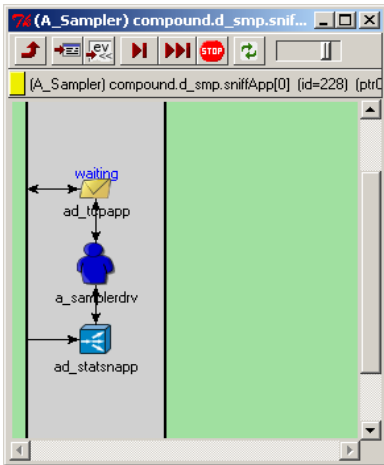


Fig. 2. General structure of agent “sampler”

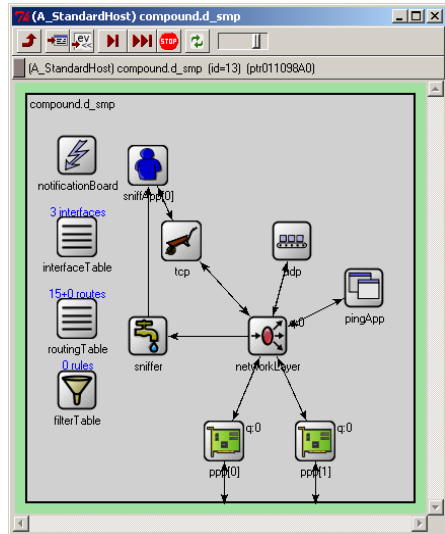


Fig. 3. Deployment of agent “sampler” into the environment

The example of *multi-window user interface of the simulation environment* is depicted in figure 4. At the basic window of visualization (figure 4, at upper right), a simulated computer network is displayed.

The window for simulation management (at the bottom right of figure 4) allows looking through and changing simulation parameters. It is important that you can see the events which are valuable for understanding attack and defense mechanisms on a time scale. The time scale is depicted above windows with the events description.

Corresponding status windows show the current status of agent teams (see the defense team status window at the upper left of figure 4). It is possible to open different windows which characterize functioning (the statistical data) of particular hosts, protocols and agents (see these windows at the bottom left of figure 4).

The example of hierarchy of simulated objects is represented in figure 5 (from left to right there are showed the nested objects “network”, “host”, “agent”). During investigation one can move from one hierarchy level to another and analyze functioning parameters of various objects.

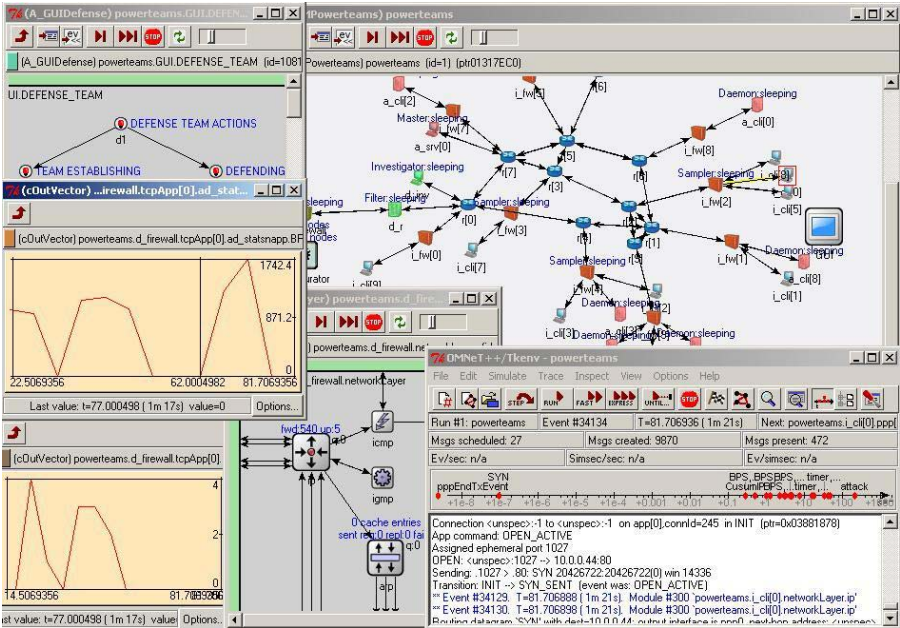


Fig. 4. Common representation of the simulation environment

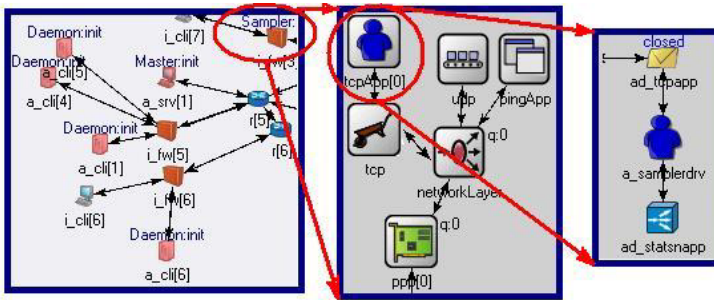



Fig. 5. Example of object hierarchy: “network” → “host” → “agent”

At the basic window of visualization (figure 6), a simulated computer network is displayed. The network represents a set of hosts and channels. Hosts can fulfill different functionality depending on their parameters or a set of internal modules. The routers are depicted with the sign “”. Attack agents are deployed on the hosts marked with red color. Defense agents are located on the hosts marked with green color. Above the colored hosts there are strings indicating the corresponding state of deployed agents. The other hosts are standard hosts that generate normal traffic.

Each network for simulation consists of three types of sub-networks: (1) the subnet of defense where the defense team is deployed; (2) the intermediate subnet where the standard hosts are deployed. They produce the generic traffic including the traffic to defended host; (3) the subnet of attack where the attack team is deployed.

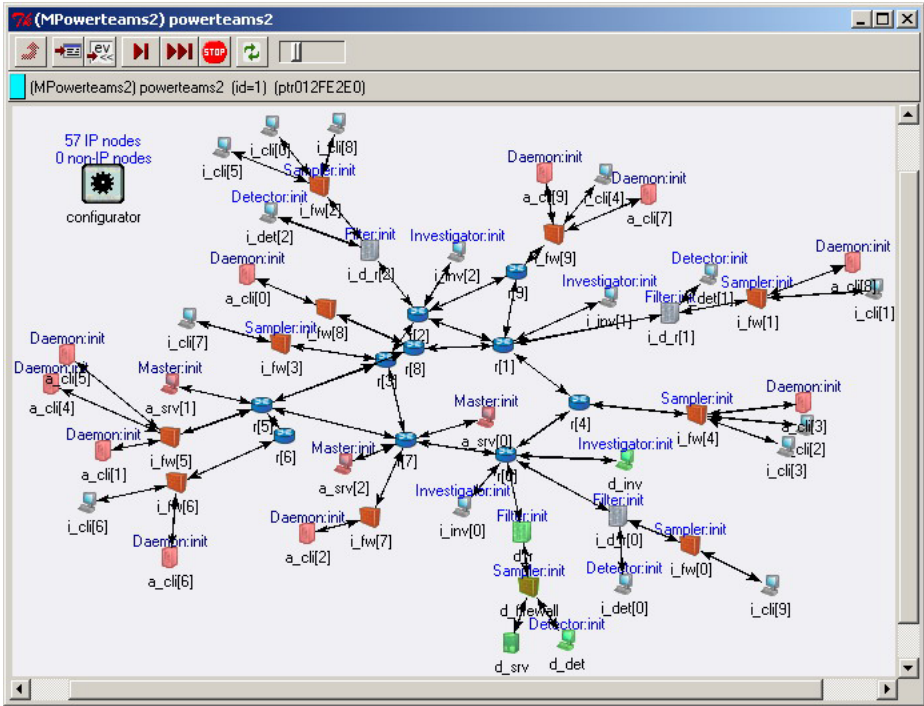


Fig. 6. Example of computer network for simulation

The subnet of defense as a rule includes at least five hosts. The following agents are deployed on the first four hosts: detector, sampler, filter and investigator. The web-server which is under defense is deployed on the fifth hosts. The agents and the web-server are the applications installed on corresponding hosts. The IP addresses are being set automatically. It is necessary to set the other application parameters. Web-server is deployed on the host *d_srv*. The interaction port and the answer delay must be set. Detector is deployed on the host *d_det*. The following parameters are used for detector: the defended host IP address, the port for team interaction, the interval for sensor inquiry, and the maximum allowed data-rate to server (BPS, bit per second). Sampler is placed on the host *d_firewall* (on the entrance to the server subnet). Filter is installed on the host *d_r* (router). Investigator is deployed on the host *d_inv*. For each of the last three agents, the private port, the IP address of detector and the port for team interaction must be determined.

The intermediate subnet includes *N* hosts *i_cli[...]* with generic clients. They are connected by the router *i_r*. The number of hosts *N* is the simulation parameter which can be set. The following parameters of clients must be specified: IP-address and port of server, the time of work start, the quantity and size of requests while connecting to server, the size of reply and the time of reply preparation, the idle interval.

The subnet of attack consists of *M* hosts *i_cli[...]* with daemons and one host with master. The number of hosts *M* must be set. Master has the following parameters: port for team interaction, IP-address and port of attack target, the time of start of attack

and its rate (measured in packets per second). Daemon has the following parameters: the port, masters' IP-address and port for team interaction.

To simulate the Internet processes (including DDoS defense and attack mechanisms), we needed the models of single hosts and topology as the representation of the way these hosts are connected. In relevant publications (e.g. [1], [23]) the Internet is represented as a graph built due some analytical dependencies. One of the main parameters to build the correct graph is node degree k . It is the amount of nodes with which the given node is connected. The average node degree is defined by the following formulae: $k = 2m/n$, where m is the amount of connections, and n is the amount of nodes in the network.

The network topology which is similar to Internet can be built on the basis of node degree. The function that can determine k for every node in network is needed. [13] summarizes the data on investigating this function. The probability density function (PDF) of node degree is built upon the data from the distributed sensors ("skitters"), BGP tables and WHOIS [13]. PDF of k for the Internet is similar to the function $f(k) = ck^{-\gamma}$, and the values of k are bounded in the following way [5]:

$$k_{\min} \leq k \leq k^{1/(\gamma-1)}.$$

The graph that represents the network topology is built in the following way [3]. There is chosen the amount of nodes n . It is generated the random value k_i on the basis of distribution $f(k)$ for every node (the sum of k_i must be even). Then every node i from the set is connected with the other k_i randomly chosen node. There are the other ways to build the random graph. The generation due to clustering method and joint degree distribution are more precise [13]. The basic network that is used for simulations in the developed environment is built in compliance with described algorithm and PDF (*). The value $\gamma = 2.25$ is borrowed from [13]. On the basis of experimental data the minimum node degree is 2.

6 Simulation Scenario Examples

The attack parameters used in the experiments represented in the paper are as follows (see section 4): Victim type – host (server that provides some service); Attack type – brute-force; Impact on the victim – disruptive; Attack rate dynamics – constant, variable; Agents' set permanency – constant, variable; Possibility of exposure – discoverable filterable attack; Source addresses validity – valid (real), spoofed: random, subnet; Degree of automation – semi-automatic with direct communication.

In the experiments considered in the paper the following defense parameters were used (see section 4): Deployment location – intermediate, defended subnets; Mechanism of cooperation – centralized; Covered defense stages – attack prevention, attack detection, attack source detection, attack counteraction; Attack detection technique – anomaly detection (Hop-count Filtering (HCF), Source IP address monitoring (SIPM), Bit per Second (BPS)); Attack source detection technique – can detect when source address is not spoofed; Attack prevention technique – packet filtering; Technique for gathering of model data – learning; Determination of deviation from model

data: thresholds (HCF, BPS), determination of fluctuation in probabilistic traffic parameter (SIPM).

Learning mode. The main task of learning mode is to create the model of generic traffic for the given network. The clients send the requests to the server and it replies. At this time sampler analyses requests and uses them to form the models of normal traffic and other parameters. During the learning it is possible to watch the change of traffic models (see figures 7-11).

Figure 7 represents the list of hosts that sent requests to server and hops to them after 300 seconds of learning and the time of last request. As mentioned above the hop count is calculated on the basis of TTL packet field.

Figure 8 depicts the change of new addresses amount for sampler during first 300 seconds of learning. One can see that in the beginning when clients requested server at the first time there were many new addresses (the maximum is 6 addresses, the time interval is 10 seconds, and the shift is 3 seconds). The last unknown address appeared in the region of 100 first seconds. At least, when all clients have requested the server there were no new addresses.

Figure 9 shows the list of clients requested the server and considered as legitimate after first 300 seconds of learning. One can see here that in the interval between 0 and 50 seconds there were many new addresses.

Figure 10 represents the graph of change of maximum BPS (for interval 10 seconds and shift 3 seconds) after 300 seconds from the beginning of learning. The maximum value was 1742.4 bit/s and was recorded in the area of 100 seconds. One can see also the values of BPS for clients that requested server in the current time interval.

Figure 11 depicts the values of transmitted bits for every client that requested server in the interval of 10 seconds.

Decision making and acting. Simulation scenario is realized on the same configuration as was used during learning. The only difference is that the attack team is engaged. Attack team initial parameters are as follows: target_ip="d_srv" (target of attack is server d_srv); target_port="2001" (target port); t_ddos=300 (time of attack start); attack_rate=5 (intensity of attack in packets per second); ip_spoofing="no" (no IP spoofing is used).

Figure 12 represents the graphs of channel throughput (bits/s to seconds) on the entrance to the defended network before (dashed line) and after (firm line) filter.

After modeling start the clients begin to send requests to the server and it replies. This is the way the generation of generic network traffic takes place (figure 12, interval 0 – 300 seconds). The formation of defense team occurs after some time from start. Investigator, sampler and filter connect to detector and send it the messages that they are alive and ready to work. Detector stores this information. The attack team is formed in the same way. Daemons connect to master and report their status. After establishing the defense team begins to function. Sampler collects traffic data and compares it with the model data acquired during learning mode. The addresses that are the source of anomalies are sent to detector every n seconds (in this scenario $n=60$). Detector makes the decision about the attack and sends to filter and investigator the addresses of suspicious hosts.

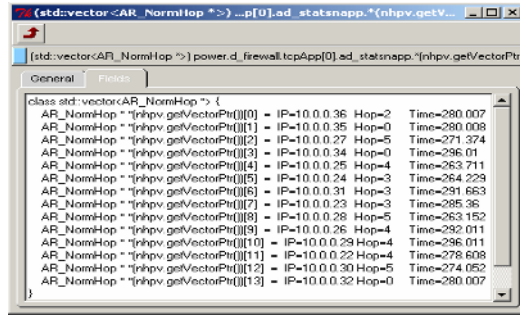


Fig. 7. List of hosts that sent requests to server and hops to them after 300 sec of learning

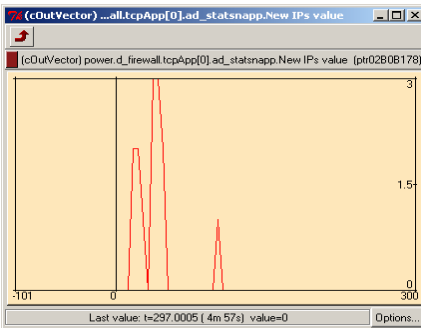


Fig. 8. Change of new IP addresses amount

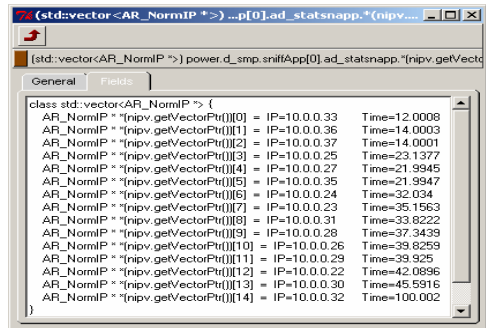


Fig. 9. List of clients requested server and considered as legitimate after 300 sec of learning

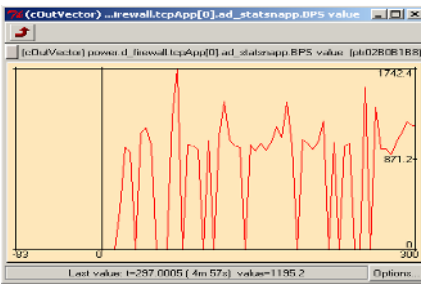


Fig. 10. Change of BPS parameter

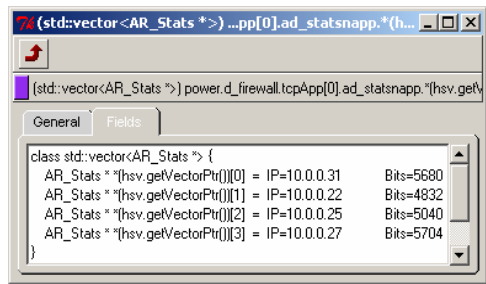


Fig. 11. Values of transmitted bits

After 300 seconds from simulation start the attack team begins attack actions. Master examines all daemons that it knows. Then it sends the command of attack to all workable daemons. This command includes address and port of attack target, intensity (distributed among daemons) and the method IP spoofing. In this case they are: target – d_srv, port – 2001, intensity of attack for every daemon (calculated as intensity divided by the number of daemons) $5/10=0.5$, spoofing “no” (no IP spoofing). When

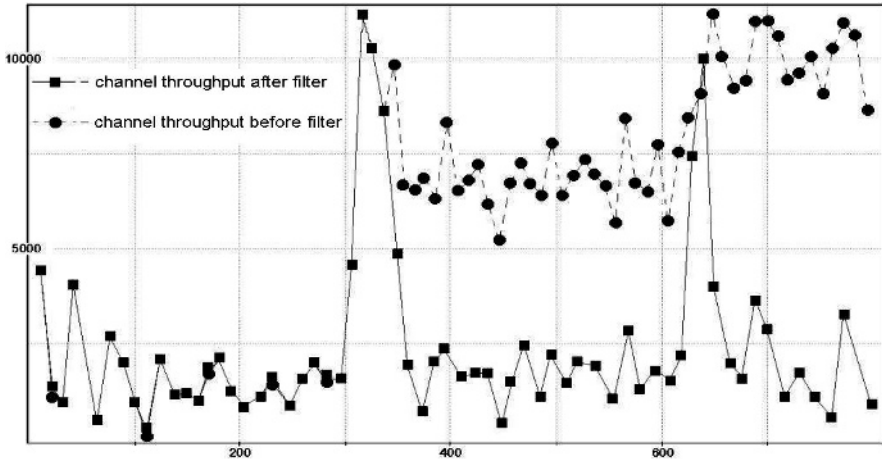


Fig. 12. Graphs of channel throughput

daemons receive the command they begin to send the attack packets (figure 12, timestamp 300 seconds).

After a while, sampler determines the suspicious hosts with the use of BPS method. The BPS parameter of these hosts exceeds normal value. Detector receives the addresses of these hosts from sampler and sends them to filter and investigator. Filter sets the filtering rules and the packets from the given hosts begin being dropped (figure 12, timestamps 400 – 600 seconds, firm graph).

Investigator tries to inspect the given hosts and to defeat the attack agents deployed there. It succeeds in defeating of four daemons. The string “defeated” appears above the defeated agent in the window of network structure. However the other daemons continue the attack (figure 12, after 400 seconds, dashed graph).

Master examines daemons next time 600 seconds after simulation has started. It does not succeed to connect with all daemons as some of them were defeated by investigator. Master makes the decision to redistribute the intensity of attack to keep the overall intensity on the given level. Also it decides to change the method of IP spoofing to complicate the detection and defeating of attack agents by defense team. Master sends to alive daemons the command: target – `d_srv`, target port – 2001, intensity – $5/(10-4)=0.83$, IP spoofing method – “random”. When daemons receive the command they continue to send the attack packets having applied the new parameters (figure 12, timestamp 600 seconds).

Detector sees that the input channel throughput has noticeably lowered since the traffic from attack team has raised (figure 12, after 600 seconds). Detector does not receive the anomaly report from sampler though. This is because the method BPS used by sampler does not work fine when attacker changes the sender address in every packet. That is the reason detector fails to confront some address with the big traffic. Therefore detector decides to apply another DDoS defense method – SIPM. The large amount of new IP addresses for sampler will lead to attack detection and dropping malicious packets. This method however does not allow tracing the source

of attack directly, and investigator will fail to defeat attack agents. But the attack packets will be filtered and the traffic in the subnet of defended host will return to normal state dropped (figure 12, timestamps 400 – 600 seconds, firm graph).

The following *effectiveness and efficiency parameters of different defense mechanisms* were studied during experiments: rate of dropped legitimate traffic (false positive rate); rate of admitted attack traffic (false positive rate); attack reaction time. These parameters were investigated in dependence on the following *input parameters*: network configuration (the amount of legitimate clients); attack intensity; IP address spoofing technique used in attack; internal parameters of defense mechanisms and their combinations; quantity and distribution of defense teams, etc.

7 Conclusion

The main results of the work we described in the paper consist in developing an approach to agent-based simulation of defense mechanisms against attacks and implementing the software environment DDoSSim intended for simulation of DDoS attacks and defense. The goal of the paper is not to present an investigation of new defense methods, but to show the possibilities of the simulation tool developed. One of the features of this tool is the possibility to insert new attack and defense methods and investigate them. The environment developed is written in C++ and OMNeT++. It allows imitating a wide spectrum of real life DDoS attacks and defense mechanisms.

Various experiments with this environment have been fulfilled. These experiments include the investigation of attack scenarios and protection mechanisms for the networks with different structures and security policies. One of the scenarios was demonstrated in the paper. Future work is connected with building more powerful simulation environment based on large library of DDoS attack and defense mechanisms, investigating new defense mechanisms, and conducting experiments to both evaluate computer network security of large-scale network security solutions and analyze the efficiency and effectiveness of different security policies against various attacks. The special attention will be given to cooperative defense mechanisms that are based on the deployment of defense components in various Internet subnets.

Acknowledgments. The research is supported by grant of Russian Foundation of Basic Research (№ 04-01-00167), Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract №3.2/03), Russian Science Support Foundation and by the EC as part of the POSITIF project (contract IST-2002-002314).

References

1. CAIDA. <http://www.caida.org/tools/>
2. Canonico, R., Cotroneo, D., Peluso, L., Romano, S.P., Ventre, G.: Programming routers to improve network security. Proceedings of the OPENSIG 2001 Workshop Next Generation Network Programming (2001)
3. Catanzaro, M., Boguñá, M., Pastor-Satorras, R.: Generation of uncorrelated random scale-free networks. *Physical Review E* 71, 027103 (2005)

4. Chen, S., Song, Q.: Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, Vol.16, No.7 (2005)
5. Dorogovtsev, S.N., Mendes, J.F.F.: The shortest path to complex networks. <http://arxiv.org/abs/cond-mat/0404593> (2004)
6. Fan, X., Yen, J.: Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents. *Journal of Physics of Life Reviews*, Vol. 1, No.3 (2004)
7. FIPA. <http://www.fipa.org>
8. Gil, T.M., Poletto, M.: MULTOPS: a data-structure for bandwidth attack detection. *Proceedings of 10th Usenix Security Symposium* (2001)
9. Jin, C., Wang, H, Shin, K.G.: Hop-count filtering: An effective defense against spoofed DDoS traffic. *Proceedings of the 10th ACM Conference on Computer and Communications Security* (2003)
10. Keromytis, A.D., Misra, V., Rubenstein, D.: SOS: An architecture for mitigating DDoS attacks. *Journal on Selected Areas in Communications*, Vol. 21 (2003)
11. Kuznetsov, V., Simkin, A., Sandström, H.: An evaluation of different ip traceback approaches. *Proceeding of the 4th International Conference on Information and Communications Security* (2002)
12. Li, M., Chi, C.H., Zhao, W., Jia, W.J., Long, D.Y.: Decision Analysis of Statistically Detecting Distributed Denial-of-Service Flooding Attacks. *Int. J. Information Technology and Decision Making*, Vol.2, No.3 (2003)
13. Mahadevan, P., Krioukov, D., Fomenkov, M., Huffaker, B., Dimitropoulos, X., Claffy, K., Vahdat, A.: Lessons from Three Views of the Internet Topology: Technical Report. *Cooperative Association for Internet Data Analysis (CAIDA)* (2005)
14. Mahajan, R., Bellovin, S.M., Floyd, S.: Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review*, Vol.32, No.3 (2002)
15. Mirkovic, J., Martin, J., Reiher, P.: A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms, Technical report #020018. *University of California, Los Angeles* (2002).
16. Mirkovic, J., Dietrich, S., Dittrich, D., Reiher, P.: *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR (2004)
17. Mirkovic, J., Robinson, M., Reiher, P., Oikonomou, G.: *Distributed Defense Against DDOS Attacks*. University of Delaware. Technical Report CIS-TR-2005-02 (2005)
18. OMNeT++ homepage. <http://www.omnetpp.org/>
19. Papadopoulos, C., Lindell, R., Mehlinger, I., Hussain, A., Govindan, R.: Cossack: Coordinated suppression of simultaneous attacks. *Proceedings of DISCEX III* (2003)
20. Park, K., Lee, H.: On the Effectiveness of Route-based Packet Filtering For Distributed DoS Attack Prevention in Power-law Internet. *Proceedings ACM SIGCOMM* (2001)
21. Peng, T., Christopher, L., Kotagiri, R.: Protection from Distributed Denial of Service Attack Using History-based IP Filtering. *IEEE International Conference on Communications* (2003)
22. Peng, T., Leckie, C., Kotagiri, R.: Proactively Detecting DDoS Attack Using Source IP Address Monitoring, *Networking 2004*, Athens, Greece (2004)
23. Route-Views Bibliography. <http://www.routeviews.org/papers/>
24. Tambe, M.: Towards flexible teamwork. *Journal of AI Research*, Vol.7 (1997)
25. Tambe, M., Pynadath, D.V.: Towards Heterogeneous Agent Teams. *Lecture Notes in Artificial Intelligence*, Vol.2086 (2001)
26. Xiang, Y., Zhou, W.: An Active Distributed Defense System to Protect Web Applications from DDoS Attacks. *The Sixth International Conference on Information Integration and Web Based Application & Services* (2004)
27. Xuan, D., Bettati, R., Zhao, W.: A gateway-based defense system for distributed dos attacks in high-speed networks. *IEEE Transactions on Systems, Man, and Cybernetics* (2002)

SNOOZE: Toward a Stateful NetwOrk prOtocol fuzZEr

Greg Banks, Marco Cova, Viktoria Felmetsger, Kevin Almeroth,
Richard Kemmerer, and Giovanni Vigna

Department of Computer Science
University of California, Santa Barbara
{nomed, marco, rusvika, almeroth, kemm, vigna}@cs.ucsb.edu

Abstract. Fuzzing is a well-known black-box approach to the security testing of applications. Fuzzing has many advantages in terms of simplicity and effectiveness over more complex, expensive testing approaches. Unfortunately, current fuzzing tools suffer from a number of limitations, and, in particular, they provide little support for the fuzzing of stateful protocols.

In this paper, we present SNOOZE, a tool for building flexible, security-oriented, network protocol fuzzers. SNOOZE implements a stateful fuzzing approach that can be used to effectively identify security flaws in network protocol implementations. SNOOZE allows a tester to describe the stateful operation of a protocol and the messages that need to be generated in each state. In addition, SNOOZE provides attack-specific fuzzing primitives that allow a tester to focus on specific vulnerability classes. We used an initial prototype of the SNOOZE tool to test programs that implement the SIP protocol, with promising results. SNOOZE supported the creation of sophisticated fuzzing scenarios that were able to expose real-world bugs in the programs analyzed.

Keywords: Stateful Fuzzing, Network Protocols, Security Testing.

1 Introduction

Security is a critical factor in today's networked world. The complexity of many network protocols combined with time-to-deliver constraints imposed on developers and improper or insecure coding practices make errors inevitable. As a result, new vulnerabilities in network-based applications are found and advertised on a daily basis. The impact of vulnerability exploitation can be severe, and, in addition, the cost of correcting errors after a system has been deployed can be very high. Therefore, we need effective methods and tools to identify bugs in network-based applications before they are deployed on live networks.

One of the methodologies used to carry out this task is *fuzzing* [1,2,3]. Fuzzing is a form of black-box testing whose basic idea is to provide a system with unexpected, random, or faulty inputs, which expose corner cases not considered during implementation.

Fuzzing has a number of advantages over other testing techniques, such as manual code review, static analysis, and model checking. First, fuzzing can be

applied to programs whose source code is not available. Second, fuzzing is largely independent of the internal complexity of the examined system, overcoming practical limits that prevent other testing methods (e.g., static analysis) from being able to operate on large applications. Being completely independent of the tested program's internals, the same fuzzing tool can be reused to test similar programs regardless of the language in which they are implemented. Finally, bugs found with fuzzing are reachable through user input, and, as a consequence, are exploitable.

A number of tools make use of fuzzing as a technique to test systems. They generally present limitations that hinder their wider and more effective use. In many cases, the means available to inject faults in the generated input are restrictive and do not include methods to specifically generate inputs that would likely trigger well-known, target-specific attacks. Furthermore, support for testing complex, stateful protocols is generally lacking; thus, requiring the tester to manually bring the system to the desired state before starting the actual test. Finally, the language adopted to describe how fuzzing should be performed is often very primitive, and, as a consequence, the activity of specifying fuzzing tests can require significant effort.

In this paper, we propose SNOOZE, a tool for building flexible, security-oriented, network protocol fuzzers. In SNOOZE we try to integrate the strengths of existing fuzzing tools, while correcting the limitations discussed above.

We have built a prototype of SNOOZE, and we used it to perform fuzzing of network applications that implement the Session Initiation Protocol (SIP) [4]. We decided to focus on SIP-based applications for several reasons. First, SIP is one of the core protocols of the VoIP infrastructure, which is becoming increasingly popular. Second, there are many competing implementations of SIP, some of which are not completely stable and have not undergone a full security assessment. Finally, SIP is a fairly complex, stateful protocol with many nuances and details that complicate its implementation and, therefore, its testing.

The contributions of this work are twofold:

1. We identify the requirements for a class of sophisticated fuzzers that can be used to test complex protocols.
2. We present the design and discuss the prototype implementation of a fuzzing approach that supports the testing of stateful protocols.

Our approach allows testers to build better fuzzers to evaluate more easily and more thoroughly the security strengths and weaknesses of complex, stateful protocol implementations. As a result, our tool can be used to limit the number and severity of vulnerabilities in deployed systems. We tested our tool on three real-world implementations of the SIP protocol, and we were able to identify previously unknown vulnerabilities.

The rest of the paper is organized as follows. In the next section we discuss the fundamental characteristics of fuzzing. In Section 3 we review related work. In Section 4 we present our approach and analyze the first prototype of SNOOZE. The evaluation of our tool is presented in Section 5. Section 6 concludes and discusses future work.

2 Background

Fuzzing is a black-box approach to testing the security properties of a software component. Fuzzing operates on the input and output of a component without requiring any knowledge of its internal working. The technique of fuzzing aims to expose flaws in applications by exercising them with invalid inputs.

Fuzzing requires three basic operations: generating random or unexpected input that could lead the application under test into an invalid state; injecting this input into the application; and, finally, observing whether the input causes the application to fail. Fuzzing relies on two fundamental assumptions:

1. A significant part of the faults contained in an application can be triggered through a limited number of input sources controlled by the user.
2. The execution of a faulty portion of an application manifests itself in visible ways, e.g., by producing unexpected output, crashing the application, or making it unresponsive.

This approach is different from white-box techniques, such as static analysis and model checking, where an explicit model of the tested application, or of some of its properties, is built and validated for correctness.

Fuzzing, unlike many white-box approaches, is not complete in the sense that it is not guaranteed to expose all faults in a program. On the other hand, all flaws found through fuzzing are guaranteed to correspond to some bug in the tested code, and, therefore, fuzzing can be considered as sound.

In general, there are two orthogonal strategies for creating faulty input for an application: generation and mutation. The generation strategy uses a formal specification of the input accepted by the tested system to generate a set of valid input values. These values are then modified by applying fuzzing primitives to obtain faulty test data. Mutation, on the other hand, relies on a set of valid input values (e.g., extracted from normal sessions), which, as before, are modified using fuzzing primitives. Generation requires that a formal specification of input values be available, but it is capable of generating all valid input. The efficacy of mutation, instead, is critically dependent on the completeness of the input set that is used. However, the generated input is generally more tractable and can focus on a specific area of weakness or type of flaw.

Fuzzers can also be differentiated on the basis of their level of understanding of input semantics. More sophisticated fuzzers automatically take into account rules constraining various parts of the input. For example, the value of some input parts may be dependent on characteristics of the whole input (e.g., checksums or content length fields), while other fields may be required to be encoded in particular formats (e.g., encrypted). Less sophisticated fuzzers leave the burden of taking care of these aspects to the user.

Fuzzers also differentiate themselves in the heuristics implemented to fuzz input and in their flexibility of use. Heuristics can be based on data types (e.g., for integer types, they may test boundary conditions such as large or small numbers, zero, and negative values) or on the expected vulnerability nature (e.g., SQL injection or format string). The complexity of applying fuzzing heuristics

can range from the invocation of a function call to the modification of an input grammar.

In some scenarios, faults in a system can only be reached after performing several intermediate steps, bringing the system to a certain state. For example, it might be necessary to perform a login step before gaining access to the application functionality that needs to be tested. Stateful fuzzers have knowledge of the system's state machine and are able to perform actions that differ depending on the current state. Stateless fuzzers, however, regard each input as completely independent. This is a substantial limitation and the main motivation behind the development of our stateful fuzzer.

3 Related Work

Fuzzing has been long used as a testing technique in areas not directly related to security (e.g., for reliability and fault tolerance assessment). One of the first uses of fuzzing is described by Miller et al. in [1]. In this paper the authors tested several standard UNIX utilities by giving them random input. The same methodology was used in later tests on the same applications [2] and on Windows [3] and MacOS [5] applications. All these tests make use of very simple fuzzing techniques, based on the generation of large chunks of random data, and have limited support for the testing of network protocol implementations.

Similar approaches proved useful when testing large, heterogeneous and complex systems, such as hardware components, real-time systems, and distributed applications. Loki, ORCHESTRA, and NFTAPE are significant examples of fault injectors¹ specifically designed for their environments. Loki allows one to inject faults in a distributed system on the basis of a partial view of the global state [6]. ORCHESTRA is a fault injection framework for distributed systems in which faults are specified as Tcl scripts, which are injected in a layered protocol stack [7]. NFTAPE adds support for multiple fault models and fault injection methods [8]. These approaches are very interesting but their focus is not on security.

More recently, fuzzing has been applied to the testing of web services. For example, one effort describes a fuzzer for web form-based services [9], while another presents dependability tests of SOAP components [10]. WSDigger is an open source tool for black-box testing of web services that takes the WSDL file of a web service as an input and tests the service with a specially crafted payload [11]. While the general ideas proposed in these works are probably applicable to different domains, they propose tools that are restricted to the testing of web services.

There are a number of tools that specifically target network protocols. The most representative of this class of fuzzers are SPIKE [12] and PROTOS [13]. The former, developed by Dave Aitel, is a framework which provides an API and set of tools to aid in the creation of network protocol fuzzers in C. In

¹ Fuzzing is usually considered to be a variant of the fault injection approach which uses randomized input.

SPIKE, a protocol packet is divided into a set of blocks, each of which can be fuzzed independently and automatically. Any change in a block size caused by a fuzzing transformation is handled automatically by SPIKE. However, the block abstraction provided by SPIKE is fairly low-level and does not allow one to easily model stateful protocols and complex messages, and their dependencies. In addition, because SPIKE-based fuzzers have to be implemented in C, their development can be effort-intensive and more complex than when using higher-level languages.

PROTOS, which was developed by the Oulu University Secure Programming Group, unlike SPIKE, does not provide an API for building custom fuzzers. Instead, it provides reusable test suites consisting of carefully crafted protocol-specific messages. Unlike other fuzzers that just send random input to a target system, PROTOS strives to generate input more intelligently by starting from the formal specification of a protocol and then using fuzzing values to generate faulty inputs. These inputs are based on heuristics that focus on triggering specific vulnerabilities, such as format string vulnerabilities and buffer overflows. The PROTOS approach has proven to be very effective: in 2002 it led to the discovery of many vulnerabilities in implementations of the Simple Network Management Protocol [14]. However, PROTOS does not provide fuzzing primitives or the ability to modify test cases without changing the protocol grammar itself, which can be a non-trivial task. Finally, it is difficult to completely evaluate because the engine used to generate test cases is not publicly available.

The goal of our project is to create a fuzzing tool that incorporates the best features of the existing fuzzers and, in addition, supports the creation of *stateful* protocol fuzzers. In the next section, we present the architecture of our fuzzing tool, which we call SNOOZE.

4 Architecture

SNOOZE is an extensible tool for the development of stateful network protocol fuzzers. It consists of the *Fault Injector*, the *Traffic Generator*, the *Protocol Specification Parser*, the *Interpreter*, the *State Machine Engine*, and the *Monitor*. Figure 1 shows the high-level architecture of SNOOZE.

The Interpreter is responsible for running the fuzzing tests. It takes as input a set of protocol specifications, a set of user-defined fuzzing scenarios, and a module implementing scenario primitives. A protocol specification defines the general characteristics of a protocol. These characteristics include, but are not limited to, the protocol type (e.g., whether it is binary or character based), the general format of header fields, the syntax of messages that can be exchanged in the protocol, and the allowed message flows (i.e., a state machine). The specification language is XML-based.

While SNOOZE has a number of protocol specifications included, new specifications can easily be added as needed. In addition, these specifications need only be written once, and they then can be reused to write testing scenarios. The Parser parses a protocol specification and makes it available to other parts of the tool.

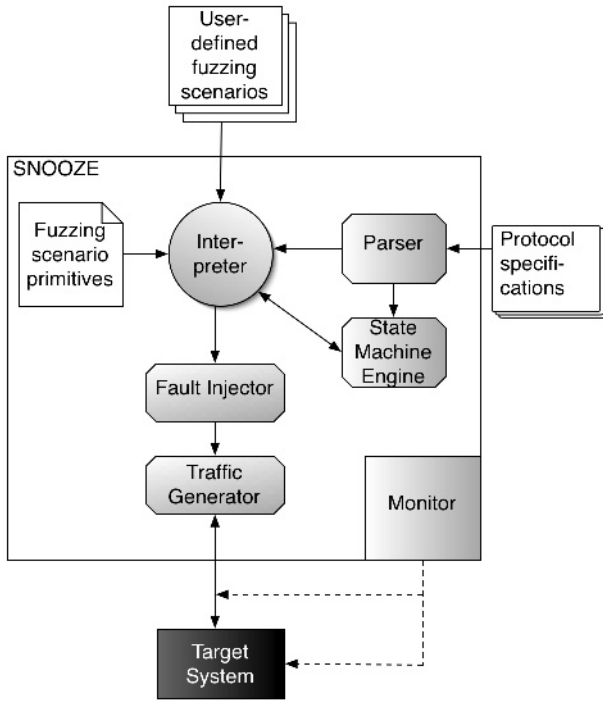


Fig. 1. Main components of SNOOZE

An example of a protocol specification is presented in Figure 2, which defines the syntax of the SIP INVITE message. In a protocol specification, each message is defined by a `<msg-rule>` element. Each `<msg-rule>` element consists of `<build-rule>` elements, which reference `<rule>` elements. The `<rule>` element either contains references to other building rules or specifies a default value for the corresponding message field. More specifically, in the example in Figure 2, the element `<build-rule id="SIP-Version"/>` specifies that each SIP INVITE message is required to contain a `SIP-Version` field, the syntax of which is defined by the `<rule>` element with ID `SIP-Version`. The default value for the `SIP-Version` field in this case is the string `SIP/2.0` concatenated with a value generated by the `CRLF` rule.

The default values assigned to fields are subject to change through the use of the mutation primitives described later. This makes it possible to modify the values of fields and to insert additional and user-defined fields into the message generated from the specification.

The dynamic aspects of a protocol, i.e., the valid sequences of exchanged messages, are specified with a state transition diagram. Each state represents a different step in the evolution of a conversation between two end-points: one state, for example, could record the fact that an INVITE message has been sent but the corresponding acknowledgment has not been received yet. The rules of

a protocol dictate the allowable transitions from one state to the other. For example, a rule would describe that when a CANCEL message is received the system should transition back to the initial state. Transitions are guarded by a condition that specifies which events can trigger the transition. Events can be the reception or the transmission of a message with specific values in determined message fields. Figure 3 shows a fragment of the specification of the SIP state diagram for a client user-agent.

```

<protocol type="ascii">
  <msg-rule id="INVITE">
    <build-rule id="INVITEm"/>
    <build-rule id="Request-URI"/>
    <build-rule id="SIP-Version"/>
    <build-rule id="Via"/>
    <build-rule id="Max-Forwards"/>
    <build-rule id="From"/>
    <build-rule id="To"/>
    <build-rule id="Call-ID"/>
    <build-rule id="CSeq"/>
    <build-rule id="Contact"/>
    <build-rule option="optional" max="inf" id="message-header"/>
    <build-rule id="Content-Length"/>
    <build-rule id="CRLF"/>
    <build-rule option="optional" max="1" id="message-body"/>
  </msg-rule>
  ...
  <rule id="SIP-Version">
    <field type="string">SIP/2.0</field>
    <build-rule id="CRLF"/>
  </rule>
  ...
</protocol>

```

Fig. 2. Part of the specification of the SIP INVITE message

In the current implementation of our tool, protocol specifications are manually extracted from standards, such as Request for Comments (RFC) documents, and can describe a protocol’s features with the level of detail desired by the user. In addition, the specifications can be used by the user to define default values to be used for the various protocol fields.

Scenario primitives are the basic operations that are available for a user to test a system; that is, they are the building blocks to derive test “drivers”. Currently, scenario primitives include mechanisms to build messages according to a protocol description, to send and wait for messages, to fuzz specific fields in a message, and to explore and leverage the state information available for a stateful protocol. Some of the available primitives are shown in Table 1.

The Fault Injector component allows a user to manipulate “normal” messages of a protocol in ways that, ideally, will cause faults in the target implementation.

```

<graph xmlns="http://www.martin-loetzsch.de/DOTML" id="SIP">
  <!-- states -->
  <node id="Start" root="true"/>
  <node id="Invite.Calling"/>
  <node id="Invite.Proceeding"/>
  <node id="Invite.Completed"/>
  <node id="Invite.CompletedAck"/>
  <node id="Terminated"/>
  ...

  <!-- transitions -->
  <edge from="Start" to="Invite.Calling">
    <send-message protocol="SIP" type="INVITE"/>
  </edge>
  <edge from="Invite.Calling" to="Invite.Proceeding">
    <recv-message protocol="SIP" type="RESPONSE">
      <field name="code" value="1?"/>
    </recv-message>
  </edge>

  ...
</graph>

```

Fig. 3. Part of the specification of the SIP state diagram

The current prototype includes a set of functions that can be used to fuzz string and integer fields in a scenario. The fuzzing functions implement various heuristics based on the testing of boundary conditions, such as very long strings, large numbers, or exploit inputs for common vulnerabilities such as SQL or command injection.

A fuzzing scenario encodes the fuzzing activity to be performed. A scenario uses the protocol specifications, scenario primitives, and the fuzzing module described above to build messages appropriate for a target protocol by fuzzing some of their fields and sending them to the target system. In the current implementation of our tool, a fuzzing scenario is a Python script that makes use of SNOOZE components and is run by the standard Python interpreter.

Figure 4 shows a complete, albeit simple, fuzzing scenario. In this scenario, we specify that we want to use SIP over UDP. We then build a SIP INVITE message with default values for every required field. After the INVITE message is built automatically by the SNOOZE engine, we set the `Request-URI` and `To` fields to some fixed value and specify that we want the `From` field to be fuzzed with values that are likely to expose an SQL injection vulnerability. The message is sent ten times using a loop. For each iteration of the loop, any piece of the `SnoozeMessage` that should be fuzzed, in this case, part of the `From` field, will contain a new fuzzed value that is automatically generated by the Fault Injector. Recall that the Fault Injector is responsible for performing fuzzing transformations on the data stored in a generic type (e.g., `SnoozeString`) as specified in that generic

Table 1. The SNOOZE primitives

Name	Description
snoozeUse	Parses the specification of the provided protocol
snoozeOpen	Opens a session with the given host and performs any required initialization
snoozeClose	Closes the given session and performs cleanup
SnoozeMessage	A class modeling protocol-independent messages
setField	Method of SnoozeMessage that allows one to set a field in a message to a given value
snoozeSend	Sends a message
snoozeExpect	Waits for a message
SnoozeString	Generic string type used in SnoozeMessage
SnoozeInt8	Generic eight bit integer type used in SnoozeMessage
SnoozeInt16	Generic sixteen bit integer type used in SnoozeMessage
SnoozeInt32	Generic thirty-two bit integer type used in SnoozeMessage
SnoozeInt64	Generic sixty-four bit integer type used in SnoozeMessage
fuzz_string_repeat	Fuzzes a field repeating a given pattern multiple times
fuzz_string_binary	Fuzzes a field inserting binary content
fuzz_string_x86nop	Fuzzes a field inserting x86 NOP instructions
fuzz_string_sql_inj	Fuzzes a field inserting strings likely to expose an SQL injection vulnerability
fuzz_string_sh_inj	Fuzzes a field inserting strings likely to expose a shell command injection vulnerability
fuzz_terminator	Fuzzes a field inserting a field terminator string
fuzz_intX_usig	Fuzzes a field inserting unsigned integer values. There exist versions for 8, 16, 32 and 64 bits integers
fuzz_intX_sig	Fuzzes a field inserting signed integer values. There exist versions for 8, 16, 32 and 64 bits integers
getValidSendMsgs	Returns the set of messages that may be validly sent in the current state of the protocol
getInvalidSendMsgs	Returns the set of messages that cannot be validly sent in the current state of the protocol
getValidReceiveMsgs	Returns the set of messages that may be validly received in the current state of the protocol
getInvalidReceiveMsgs	Returns the set of messages that cannot be validly received in the current state of the protocol
getCurrentState	Returns an object holding information about the current state of the protocol

type's constructor. At the end of the scenario, the session is closed. Figure 5 shows a selection of the messages generated by this scenario.

Figure 6 shows the use of some of the state-related primitives. As before, from the initial state, an INVITE message is sent. Since this represents a valid transition, the State Machine Engine updates the current state. The scenario, then, sends all messages that are not supposed to be sent from the current state and waits for a message. The scenario at this point could, for example, check the received packet to determine whether the invalid messages caused an unexpected transition in the implementation under test.

```

from snooze_scenario_primitives import *
from snooze_types import *

# fuzz SIP over UDP (the network profile)
profile = snoozeUse('SIP', 'udp')

host = '127.0.0.1'
port = 5060

sd = snoozeOpen(host, port, profile)

# build an INVITE message
m = SnoozeMessage('SIP', 'INVITE')
# modify default values of some fields
m.setField('Request-URI', [
    SnoozeString('ru', 'sip:test@' + host + ':' + str(port) + ' ')])
m.setField('To', [SnoozeString('tn', 'To: '),
    SnoozeString('tv', 'sip:test@' + host), SnoozeString('fe', '\r\n')])
m.setField('From', [SnoozeString('fn', 'From: '),
    SnoozeString('fr', 'sip:'),
    SnoozeString('ff', 'A', fuzz_string_sql_inj),
    SnoozeString('fv', '@' + host), SnoozeString('fe', '\r\n')])

for i in range(10):
    snoozeSend(sd, m)
snoozeClose(sd)

```

Fig. 4. An example fuzzing scenario

The `SnoozeExpect` primitive provides a mechanism to wait for messages, based on what type of protocol is being fuzzed (e.g., text or binary), the type of message, and the message's content. A scenario developer can then make conditional decisions based on the return value of the primitive, thereby navigating paths in the protocol state machine dynamically.

The operation of sending messages to the target system is performed by the Traffic Generator component. It receives messages created by a user scenario and transforms them into network packets while taking into account fields that need to be updated (e.g., checksums or content length fields). Then, it sends those packets to the target system.

The State Machine Engine keeps information about the state of network operations. In practice, it keeps track of transmitted and received messages, and it uses the protocol state diagram specification to check whether the messages trigger some transition from the current state to a new state.

Finally, the Monitor component analyzes the traffic data and the behavior of the target system, looking for manifestations of a fault. These manifestations include, but are not limited to, events such as a segmentation fault in the target system, a hang, an abnormal behavior, or an unexpected output that is the result of the system being put into an inconsistent state. In the

```
INVITE sip:test@127.0.0.1:5060 SIP/2.0
Via: SIP/2.0/TCP foo.cs.ucsb.edu:4040;branch=z9hG4bK74bf9
Max-Forwards: 70
From: sip:(SELECT%20*)@127.0.0.1
To: sip:test@127.0.0.1
Call-ID: UniQue1@tester.com
CSeq: 1 INVITE
Contact: <sip:whatever.com>
Content-Length: 0
```

```
INVITE sip:test@127.0.0.1:5060 SIP/2.0
Via: SIP/2.0/TCP foo.cs.ucsb.edu:4040;branch=z9hG4bK74bf9
Max-Forwards: 70
From: sip:%200R%201=1@127.0.0.1
To: sip:test@127.0.0.1
Call-ID: UniQue1@tester.com
CSeq: 1 INVITE
Contact: <sip:whatever.com>
Content-Length: 0
```

Fig. 5. An example of the messages sent when executing the scenario in Figure 4

current SNOOZE prototype, rudimentary monitoring is available. This is provided through the `snoozeExpect` primitive, which will alert the tester when either an unexpected message is received or a timeout expires without receiving any data, which usually indicates the target system has crashed or hung because of the last `snoozeSend`. In addition to this automated monitoring, manual inspection must be done on the target system to identify whether the system is “behaving” correctly when a fault does not manifest itself in the messages being exchanged between the fuzzer and the target system.

5 Evaluation

Evaluating a fuzzer’s performance is difficult. Generally, there is no direct feedback about the effectiveness of the fuzzer, other than the fact that the target system crashed or stopped functioning correctly. For this reason, the common evaluation practice is to run the fuzzer on a test suite of programs and evaluate its effectiveness based on the number of bugs found or the number of programs crashed. However, as discussed in previous sections, no conclusion can be derived about the completeness of the analysis performed through black-box testing. Therefore, an interesting extension to this basic practice would be to couple the number of bugs found with the amount of code exercised as a quantitative evaluation metric. We plan to investigate this extension in future tests as we believe that code coverage provides an estimate of how thorough the fuzzing process is. The assumption is that the more code paths that are traversed, the more potential bugs are discovered.

```

from snooze_scenario_primitives import *
from snooze_types import *

# fuzz SIP over UDP (the network profile). Enable the State Machine Engine
profile = snoozeUse('SIP', 'udp', 'client', True)
target_port = 5062
snooze_port = 5060

# open the session
sd = snoozeOpen('128.111.48.24', target_port, profile, snooze_port)

# build and send an INVITE message
m_inv = SnoozeMessage('SIP', 'INVITE', {'Content-Type': 'Content-Type'})
...[packet setup not shown]...
snoozeSend(sd, m_inv)

# send invalid messages
for msg in getInvalidSendMsgs():
    snoozeSend(sd, msg)

# wait for reply
snoozeExpect(sd)
...

```

Fig. 6. A scenario that uses state-related primitives

Qualitative metrics are also valuable. The ease of creating powerful fuzzing scenarios is a key factor in the adoption of one tool over another. In addition to providing fuzzing functionality, the ability to build simple, general-purpose clients is another metric to consider when comparing fuzzers.

Having decided on the appropriate metrics for evaluating our tool, we chose to focus our attention on the Session Initiation Protocol (SIP) [4]. SIP is an application-layer signaling protocol used to create, modify and terminate sessions with one or more participants, such as those found in Internet conferences and Internet telephone calls. Managing sessions involves multi-step operations. Consider for example the steps involved in the setup of a call: the caller sends an INVITE message to the callee; the user agent of the callee sends back a **Ring**ing status response; when the user answers the call, an **OK** message is generated; the caller replies to this message with an **ACK** message. A similar exchange of messages is required for call termination. A consequence of the statefulness of SIP is that many bugs can be exposed only by exploring states that are “deep” in the protocol state machine, i.e., states that are reachable only after exchanging a coherent series of messages with the application under test.

We chose SIP for our evaluation for several reasons. First, there are several open-source implementations available. This allowed us to assemble a set of applications to test and to investigate the problems we found by code inspection. Second, SIP is not yet fully mature. Several implementations are still not

completely RFC-compliant and most projects have been started within the last couple of years. Finally, SIP is currently being used as the signaling protocol for many popular IP telephony, chat, and video conferencing applications.

We built a testbed of different SIP implementations consisting of the following programs: Linphone 1.1.0 [15] compiled with libosip 2.2.2 [16], Kphone 4.2 [17], and SJphone 2.99a [18]. These programs are a representative set of commonly used programs that utilize SIP.

Our tests consisted of running a scenario that fuzzed different combinations of fields, using all of the fuzzing primitives that are currently implemented in SNOOZE. The scenario explores different states of the programs under test, by sending the sequence of messages `INVITE`, `CANCEL`, `ACK`. Note that, in this way, we set up a complete SIP dialog, comprising several transitions in the SIP state machine, and we can perform fuzzing at all of the traversed states. We let this scenario replay this message sequence 19,000 times using different fuzzing values. This fuzzing scenario did not cause any fatal error, e.g., crash or hang, in SJphone or Kphone. However, it found several problems in Linphone and hereafter we describe three bugs that are representative of the types of flaws that SNOOZE can expose.

The first example is a crash caused by the initial `INVITE` message in our test sequence. Linphone shows the identity of a caller by presenting the content of the `From` field of a SIP `INVITE` message. When receiving fuzzed messages, the message parsing routine in Linphone is unable to parse the `From` field and returns a `NULL` value to its caller instead of a valid pointer to the parsed content. Unfortunately, the caller routine does not check the returned value and blindly dereferences it, causing a segfault and a subsequent crash of the program. Even though this error cannot generally be used to further escalate privileges, it can be considered a denial of service attack. The bug has been acknowledged by the author of the program and corrected in the following release [19].

A second crash resulted from a malformed `ACK` message, the last in the sequence that we were playing. The message, on its own, had no effect, and the bug only manifested itself in the case where there was an open call. That is, there was state saved in the form of a dialog. This bug, similar to the previous one, results from an attempted `NULL` pointer dereference in `libosip`. In this particular iteration of the scenario, an `INVITE` message had been sent which caused Linphone to enter the `ringing` state. The subsequent `CANCEL` message for that call was then ignored by Linphone because it was being fuzzed with binary data, making it non-parsable. A new call was then attempted with another `INVITE` message, causing Linphone to save the current state. Several message sequences later, an `ACK` message was sent with no `Call-ID` field present. Linphone received the parsed message from `libosip`, recognized that it was an `ACK`, and iterated through the dialogs associated with each phone call, calling the `libosip` routine `osip_dialog_match_as_uas`. The first step of this routine is to convert the `Call-ID` field of the received message to a string and then compare it to the same field in the stored dialog. In this case, the internal routine to convert the `Call-ID` field to a string returns `-1`, indicating an error, which

`osip_dialog_match_as_uas` fails to check. The resulting NULL pointer is then passed to `strcmp` which causes the segmentation fault to occur. This bug would not have been found without the ability to drive the application to a state deep in the SIP state machine.

A third crash was related specifically to the application’s graphical interface. Although the details are not clear at this point, an improper series of messages cause debug statements of the form “Xlib: unexpected async reply” to be printed to the console. This problem is most likely caused by threading issues affecting the use of Xlib. The exact problem, however, is still to be determined.

From a qualitative point of view, SNOOZE, even in its first prototype version, has some advantages over other fuzzers. First, SNOOZE follows an object-oriented approach to the creation and manipulation of protocol messages by allowing the user to abstract away irrelevant details. This feature, coupled with the use of protocol specifications, greatly eases the task of dealing with messages. The result is that users can build valid protocol messages by simply invoking the `SnoozeMessage` constructor and message fields can later be manipulated by calling methods of the `SnoozeMessage` class. This contrasts with other fuzzers that require users to manually construct each message (e.g., SPIKE), and with those that allow users to modify messages only by manipulating the protocol grammar (e.g., PROTOS). Second, it provides a set of fuzzing methods that can be easily reused in multiple scenarios. Third, by design, it should be easy to use SNOOZE as the basis for general-purpose network clients, implemented using calls to SNOOZE primitives like `SnoozeMessage`, `snoozeSend` and `snoozeExpect`. This is not possible with fuzzing tools that only build test cases.

6 Conclusions

The complexity of current network protocols and the increasing number of attacks against protocol implementations require a stronger emphasis on the testing of programs. Not only more powerful but also more intuitive tools for assessing the security of network programs are needed.

We believe that fuzzing, i.e., injecting faults into a program by exercising it with random and unexpected input, can be a powerful testing tool. Even though fuzzing does not guarantee completeness of the analysis, it provides a practical way to quickly assess the robustness of an implementation to malicious input.

We described the initial design and implementation of SNOOZE, a tool for building flexible, security-oriented, multi-protocol network fuzzers. The first prototype of SNOOZE provides a set of primitives to flexibly generate fuzzed messages. It also provides support for stateful protocols, allowing for rapid development of fuzzing scenarios. The tool is protocol-independent and can be easily extended.

The preliminary results from using SNOOZE on a testbed of programs implementing the SIP protocol show that SNOOZE can be effectively used for finding bugs that are hidden deep in the implementation of stateful protocols. Moreover, the combination of reusable fuzzing primitives together with initial support for

stateful protocols allowed for implementation of quite complex stateful scenarios with reduced user effort.

In the future, we plan to extend SNOOZE in a number of directions. First, we plan to enhance the support for stateful protocols, particularly exploring ways to synchronize the state of the communication as seen by the fuzzer with the state of the application under test. Also, we plan to develop a GUI that will allow a scenario developer to build stateful scenarios graphically, in an intuitive manner. In addition, we intend to further evaluate SNOOZE using the code coverage metric. Finally, we will test the idea of composing SNOOZE with model checking tools to better model the exploration of the protocol state machine.

Acknowledgment

This research was supported by the Army Research Laboratory and the Army Research Office, under agreement DAAD19-01-1-0484. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

1. Miller, B.P., Fredriksen, L., So, B.: An empirical study of the reliability of UNIX utilities. *Communications of the ACM* **33**(12) (1990) 32–44
2. Miller, B.P., Koski, D., Lee, C., Maganty, V., Murthy, R., Natarajan, A., Steidl, J.: Fuzz Revisited: A Reexamination of the Reliability of UNIX Utilities and Services. Technical report, Computer Science Department, University of Wisconsin (1995)
3. Forrester, J.E., Miller, B.P.: An empirical study of the robustness of Windows NT applications using random testing. In: *Proceedings of the 4th USENIX Windows Systems Symposium*. (2000) 59–68
4. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: RFC 3261: SIP: Session Initiation Protocol (2002)
5. Miller, B.P., Cooksey, G., Moore, F.: An Empirical Study of the Robustness of MacOS Applications Using Random Testing. Technical report, Computer Science Department, University of Wisconsin (2006)
6. Cukier, M., Chandra, R., Henke, D., Pistole, J., Sanders, W.H.: Fault Injection Based on a Partial View of the Global State of a Distributed System. In: *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, Washington, DC, USA, IEEE Computer Society (1999) 168–177
7. Dawson, S., Jahanian, F., Mitton, T.: ORCHESTRA: A fault injection environment for distributed systems. Technical Report CSE-TR-318-96, University of Michigan (1996)
8. Stott, D.T., Floering, B., Kalbarczyk, Z., Iyer, R.K.: A Framework for Assessing Dependability in Distributed Systems with Lightweight Fault Injectors. In: *Proceedings of the 4th International Computer Performance and Dependability Symposium*. (2000) 91–102
9. Huang, Y.W., Huang, S.K., Lin, T.P., Tsai, C.H.: Web Application Security Assessment by Fault Injection and Behavior Monitoring. In: *Proceedings of the 12th International World Wide Web Conference*, New York, NY, USA, ACM Press (2003) 148–159

10. Looker, N., Xu, J.: Assessing the Dependability of SOAP RPC-Based Web Services by Fault Injection. In: Proceedings of the Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems. (2003)
11. Foundstone: WSDigger. (http://www.foundstone.com/resources/s3i_tools.htm)
12. Aitel, D.: The Advantages of Block-Based Protocol Analysis for Security Testing. Technical report, Immunity, Inc. (2003)
13. Kaksonen, R., Laakso, M., Takanen, A.: Software Security Assessment through Specification Mutations and Fault Injection. In: Proceedings of Communications and Multimedia Security Issues of the New Century. (2001)
14. Oulu University Secure Programming Group: PROTOS Test-Suite: c06-snmvp1. Technical report, University of Oulu, Electrical and Information Engineering (2002)
15. Linphone Project Team: Linphone: Telephony on Linux. (<http://www.linphone.org/>)
16. A. Moizard: The GNU oSIP library. (<http://www.gnu.org/software/osip/>)
17. KPhone Project Team: KPhone: a voice over internet phone. (<http://sourceforge.net/projects/kphone/>)
18. SJ Labs: SJphone. (<http://www.sjlabs.com/sjp.html>)
19. Morlat, S.: Re: [SNOOZE] remote crash of linphone-1.1.0. Personal communication to the authors (2006)

Rights Protection for Data Cubes

Jie Guo^{1,2}, Yingjiu Li¹, Robert H. Deng¹, and Kefei Chen²

¹ School of Information Systems, Singapore Management University, 178902, Singapore
{yjli, robertdeng}@smu.edu.sg

² School of Information Security Engineering, Shanghai Jiaotong University, 200030, China
{guojie, kfchen}@sjtu.edu.cn

Abstract. We propose a rights protection scheme for data cubes. The scheme embeds ownership information by modifying a set of selected cell values. The embedded message will not affect the usefulness of data cubes in the sense that the sum queries at any aggregation level are not affected. At the same time, the errors introduced to individual cell values are under control. The embedded message can be detected with a high probability even in the presence of typical data cube attacks. The proposed scheme can thus be used for protecting data cubes from piracy in an open, distributed environment.

1 Introduction

Data cube is a common data model that supports exploration of a large amount of electronic data from many different perspectives, in a multi-dimensional and multi-level manner. In many applications, valuable data cubes are provided to multiple users or customers. For example, business and government organizations often outsource valuable data cubes, such as sales patterns data, financial data, or customer information, to certain parties that are specialized in analyzing the data. For another example, data cubes of online interactions (e.g., airline reservation and scheduling portals) are usually provided for direct and interactive uses by many customers across the internet. In these applications, it is critical to protect the owner's rights so as to thwart any illegal use of data. Without appropriate rights protection, some dishonest users may copy and redistribute the data without its owner's permission.

Watermarking techniques have been frequently used for protecting data ownership. The ownership information is embedded into the target data in a way that it has no significant impact on the usefulness of the data and that a malicious user cannot destroy it without making the data less useful. When a pirated copy of data is discovered somewhere, the owner of the data can assert its ownership with a high probability by extracting the embedded information from the watermarked data. Watermarking has been extensively studied in the context of multimedia data (text, image, sound or video) [9] [5] [14] [15] [10] [18]. Since multimedia objects are ultimately watched or listened by human beings, it is critical that the embedded watermark has no significant impact on human perceptual systems. Recently, there have been growing interests in watermarking non-media data [3] such as relational databases [1], softwares [8], natural language texts [4], and sensor network streams [19]. The challenges posed in these new domains are quite different from those posed in the multimedia domain. As pointed out in [1][3],

non-media data have characteristics and operations very different from multimedia data, thus requiring different watermark algorithms to be designed.

Agrawal et al. first proposed a watermarking scheme for relational databases [2]. A private key, known only to the owner of the data, was used to determine where to embed a watermark and how to detect it. Li et al. extended Agrawal's watermarking scheme to embed user-specific information, called fingerprint, into relational databases [17]. The fingerprint can be used to identify or track whom the traitor is from a pirated copy of data. For numerical data, Sion et al. introduced a multi-bit distribution encoding scheme that can survive the linear transformation of data among other types of attacks [20]. For categorical data, Sion proposed another watermarking technique that swaps certain categorical values so as to embed watermark information [21]. Other works include Gross-Amblard's watermark framework [12], in which a set of parametric queries can be preserved within a certain level of distortion, and Bertino's hierarchical watermarking scheme [7] for preserving the privacy and ownership of outsourced medical data against generalization attacks. All these works are targeted on watermarking relational databases.

The watermarking techniques customized for relational databases cannot be directly applied to data cubes. One reason is that the structure of data cubes is different from that of databases. While most schemes for watermarking databases depend on the existence of a primary key attribute, there is no such concept in data cubes. Another reason is that data cubes are primarily used for answering sum queries regarding the aggregations of cell values at different levels. A watermarking scheme should have as less impact as possible, better no impact, on the data cube queries. In comparison, no special treatment has been made in watermarking relational databases for reducing or eliminating the errors in sum queries.

The contribution of this paper is multi-fold. Firstly, we propose the first watermarking scheme for protecting the ownership of data cubes. A data cube owner can use his private key to control all watermarking parameters. Neither original data cube nor the watermark is required in watermark detection. Secondly, we devise the concept of mini-cube and apply it in the process of watermarking. As a result, all sum queries in a watermarked data cube can be answered without any error, yet the robustness of the embedded watermark is not affected by the use of mini-cubes. Thirdly, we extend our basic scheme to improve watermarking efficiency for very large data cubes. Detailed analysis and extensive experiments are conducted for the proposed schemes in terms of watermark detectability, robustness, imperceptibility, and efficiency. Our results show that the scheme perform well in actual applications.

The rest of this paper is organized as follows. Section 2 revisits the basic model of data cubes. Section 3 presents our watermarking algorithms. Section 4 analyzes the properties of the proposed algorithms. Section 5 evaluates our watermarking technique using real data. Section 6 concludes this paper.

2 The Basic Model of Data Cubes

We follow the data cube model proposed in paper [11]. Conceptually, a cube consists of a base cuboid, surrounded by a collection of aggregation cuboids that represent the

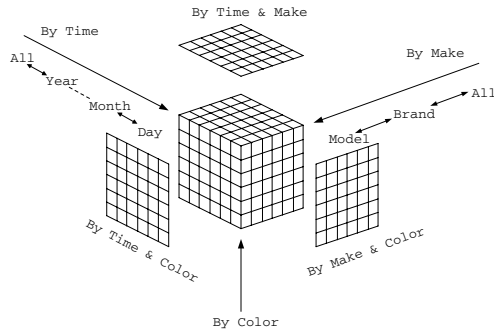


Fig. 1. An example data cube

aggregation of the base cuboid along one or more dimensions. We refer to the attribute to be aggregated as the *measure attribute*, while the other dimensions are seen as the *feature attributes*. Figure 1 gives an example of data cube from the automotive industry. The data cube is in 3 dimensions, where each cell (m, c, t) represents a combination of three feature attributes *make*, *color*, and *time*, and where a single measure attribute of the cell is *sale*. The measure attribute value of cell (m, c, t) indicates the sale of the car model m in color c on day d .

The base unit of each dimension may be aggregated to higher level units in “roll-up” operations. For example, the *time* dimension can be aggregated from the basic unit *day* to higher level units *month*, *year*, and *all*. Consequently, the cell values (i.e., car sales) are aggregated in terms of the new units in roll-up operations. If the 3-D based cuboid is rolled-up from *day* to *all* along *time* dimension, then it becomes a 2-D plane, which is labelled as “by make and color” in figure 1. The 2-D plane can be further rolled-up to a 1-D line and to a 0-D point (the 0-D point represents the grand total of the car sales for all make, all color, at all time). From higher level aggregations, one may also “drill-down” to lower level aggregations or individual cells.

Roll-up and drill-down are basic OLAP functions which can be used for knowledge discovery in decision support systems. By rolling up from the base cuboid, one can aggregate the measure attribute at various levels, thus discovering general trends of the underlying data. By drilling down to lower level aggregations or individual cells, one may discover outliers or exceptions. The interactive exploration of rolling-up and drilling down may repeat until a satisfactory understanding of the underlying data is reached. Based on the data cube model, we can design our watermarking scheme for data cubes.

3 Embedding and Detecting Watermarks

In this section, we introduce our watermarking scheme in detail. We assume that all watermarkable values in a data cube are numeric, and that small changes in a small portion of these values are acceptable. Note that the same assumption has been commonly used in watermarking relational data. It has been argued that such an assumption is supported by many applications for various types of data (e.g., parametric specifications in manufacturing industry, geological and climatic surveys, and life science data) [1].

Table 1. Notations and parameters

X_i, Y_j, Z_k	Three feature attributes of a cell
N_x, N_y, N_z	Sizes of three feature attributes in the base cuboid
η	Total number of cells in the data cube
ξ	Number of least significant bits available for watermarking in each cell
$1/\gamma$	Fraction of cells selected for embedding a watermark
ω	Number of cells selected for embedding a watermark
α	Significance level of the test for detecting a watermark
τ	Minimum number of correctly detected cells for ownership claim

Algorithm 1. Watermark embedding

```

1: for  $i = 1$  to  $N_x$ ,  $j = 1$  to  $N_y$ ,  $k = 1$  to  $N_z$  do
2:    $HMAC(i, j, k) = \mathcal{H}(\mathcal{K} \oplus opad, \mathcal{H}(\mathcal{K} \oplus ipad, X_i \circ Y_j \circ Z_k))$ 
3:    $mark(i, j, k) = HMAC(i, j, k) \bmod \gamma$ 
4: end for
5:
6: for  $i = 1$  to  $N_x$ ,  $j = 1$  to  $N_y$ ,  $k = 1$  to  $N_z$  do
7:   if ( $mark(i, j, k) = 0$ ) then // This cell is selected for marking
8:      $bp = HMAC(i, j, k) \bmod \xi$  // Mark position at this cell is  $bp^{th}$  bit
9:      $wm = HMAC(i, j, k) \bmod 2$  // Watermark allocated to this cell is  $wm$ 
10:     $bm = (d(i, j, k) \gg bp) \& 1$  // The  $bp^{th}$  LSB of the cell value is  $bm$ 
11:    if ( $bm \neq wm$ ) then // The cell value should be changed for marking
12:      set the  $bp^{th}$  least significant bit of the cell value to  $wm$ 
13:       $df =$  the difference between marked cell value and original cell value
14:       $minicube(i, j, k, df, D, mark)$  // Construct a minicube for the modified cell
15:    end if
16:  end if
17: end for

```

Without loss of generality, we present our scheme for a 3-dimensional data cube D , where each cell has three feature attributes (X, Y, Z) and one measure attribute M . The sizes of three feature attributes (i.e., the numbers of base units) are N_x, N_y, N_z , respectively. The total number of cells η is $N_x \times N_y \times N_z$. We use $d(X_i, Y_j, Z_k)$ to denote the cell value at the position (X_i, Y_j, Z_k) . For each watermarkable cell, we assume that any change in one of its ξ least significant bits is imperceptible, where ξ is a parameter in our scheme. Another parameter γ determines the number ω of watermarkable cells that are used to embed a watermark. Approximately one out of every γ values is used in watermarking (i.e., $\omega \approx \eta/\gamma$). A significance level α is used to determine how amenable the watermarking system is to the false detection of a watermark from non-watermarked data. A parameter τ is used to denote the minimum number of correctly detected cells for ownership claim. The two parameters will be further explained in Section 3.2. For ease of reference, Table 1 gives the notations that will be used in this paper.

3.1 Watermark Embedding

The procedure of watermark embedding is shown in Algorithm 1. Let HMAC be a MAC function seeded with a private key [6][16]. For each cell in a 3-D data cube, the owner of the data computes a HMAC value from the cell's feature attributes (X_i, Y_j, Z_k) using his private key \mathcal{K} . The HMAC value will be used to determine whether the cell is

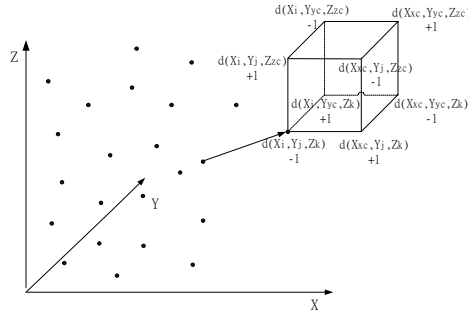


Fig. 2. An example mini-cube

selected to embed a watermark (see lines 3 and 7 in algorithm 1). On average, one out of every γ cells is selected. Because of the use of the HMAC function and the private key, only the owner of the data can determine which cells are actually selected. For each selected cell, line 8 in algorithm 1 determines a bit position among ξ least significant bits, and line 9 computes a watermark bit value which is assigned to the bit position. The watermark bit has a probability of 1/2 to be the same as the original bit in the bit position, in which case the selected cell does not change. Otherwise (see line 11), the original bit is flipped, in which case the selected cell is modified so as to embed the watermark bit.

According to our assumptions, the change made in watermarked cells is acceptable for individual values; however, it can be potentially significant to some aggregations of the cell values. In order to reduce or eliminate the accumulative errors that are introduced in watermark insertion, a mini-cube is constructed for each cell that is modified in watermarking (see line 14 in algorithm 1). Figure 2 illustrates a simple example for constructing a mini-cube. Suppose that the value of cell $d(X_i, Y_j, Z_k)$ is decremented by 1 in watermark insertion. Based on the position of $d(X_i, Y_j, Z_k)$, three other cell values $d(X_{xc}, Y_j, Z_k)$, $d(X_i, Y_{yc}, Z_k)$, and $d(X_i, Y_j, Z_{zc})$ are selected. The values of these cells are incremented by one so as to balance the deviation in any 1-D aggregation (i.e., aggregation along one feature dimension) that involves cell (X_i, Y_j, Z_k) . Similarly, three more cell values $d(X_{xc}, Y_j, Z_{zc})$, $d(X_i, Y_{yc}, Z_{zc})$, and $d(X_{xc}, Y_{yc}, Z_k)$ are decremented by one, and one last cell value $d(X_{xc}, Y_{yc}, Z_{zc})$ is incremented by one. These seven cells, which we call *balance cells*, form a mini-cube together with the watermarked cell (X_i, Y_j, Z_k) . With a mini-cube constructed, any data cube aggregation that involves at least two cells in the mini-cube remains unchanged after watermark insertion. Algorithm 2 gives the procedure for constructing a mini-cube.

A mini-cube is not constructed without a constraint. Firstly, the balance cells should not be selected from any cell that has been selected in watermark insertion so as to avoid interfering the watermark insertion and detection. Secondly, a mini-cube should be constructed in a way that most, if not all, aggregation queries would involve at least two cells in the mini-cube. To achieve this, (i) the balance cells should be selected as close to the watermarked cell as possible, and (ii) any two cells in the mini-cube should have the same attribute values *in terms of the smallest aggregation units* above the base unit (e.g., “brand” for attribute *model* and “month” for attribute *day* are such aggregation units in

Algorithm 2. Tow functions: mini-cube embedding and choosing cells for mini-cube

```

1: mini-cube(X-index  $i$ , Y-index  $j$ , Z-index  $k$ , value-difference  $df$ , data cube  $D$ , embedding position  $mark$ )
2:  $xc = 0; yc = 0; zc = 0;$  // Parameters used for deciding cell positions for mini-cube
3:  $(xc, yc, zc) = CellPosition(i, j, k, D, mark)$  // Choose cell positions for mini-cube
4: if ( $xc \neq i$ ) then // Construct mini-cube
5:    $d(xc, j, k) = d(xc, j, k) - df$ 
6:    $d(i, yc, k) = d(i, yc, k) - df$ 
7:    $d(i, j, zc) = d(i, j, zc) - df$ 
8:    $d(xc, yc, k) = d(xc, yc, k) + df$ 
9:    $d(xc, j, zc) = d(xc, j, zc) + df$ 
10:   $d(i, yc, zc) = d(i, yc, zc) + df$ 
11:   $d(xc, yc, zc) = d(xc, yc, zc) - df$ 
12: end if
13:
14: CellPosition(X-index  $i$ , Y-index  $j$ , Z-index  $k$ , data cube  $D$ , embedding position  $mark$ )
   return (X-position  $xc$ , Y-position  $yc$ , Z-position  $zc$ )
15:  $lx = round(i/\lambda_x) * \lambda_x$  // Round real value to integer
16:  $my = round(j/\lambda_y) * \lambda_y$ 
17:  $nz = round(k/\lambda_z) * \lambda_z$ 
18:  $xc = i; yc = j; zc = k;$  // Parameters used for deciding cell positions for mini-cube
19:  $SumThresh = 0$  // Value used for finding optimal mini-cube
20: for  $l = lx$  to  $lx + \lambda_x - 1$ ,  $m = my$  to  $my + \lambda_y - 1$ ,  $n = nz$  to  $nz + \lambda_z - 1$  do
21:  // Balance cells are not selected from the cells that are selected in watermark insertion
22:  if ( $(mark(l, j, k) \neq 0)$  and ( $mark(i, m, k) \neq 0$ ) and ( $mark(i, j, n) \neq 0$ ) and ( $mark(l, m, k) \neq 0$ )
   and ( $mark(l, j, n) \neq 0$ ) and ( $mark(i, m, n) \neq 0$ ) and ( $mark(l, m, n) \neq 0$ )) then
23:     $sum = |d(l, j, k)| + |d(i, m, k)| + |d(i, j, n)| + |d(l, m, k)| + |d(l, j, n)| + |d(i, m, n)| + |d(l, m, n)|$ 
24:    if ( $sum > SumThresh$ ) then
25:       $SumThresh = sum; xc = l; yc = m; zc = n;$ 
26:    end if
27:  end if
28: end for
29: return ( $xc, yc, zc$ )

```

Figure 1). We use three parameters λ_x , λ_y , and λ_z to decide how far away along each dimension to search the balance cells from a watermarked cell. These parameters can be fixed or floating for different watermarked cells. The last requirement for constructing a mini-cube is that the modification to the balance cells should be minimum. The cells with larger values are better to be selected to be balance cells, for smaller values are more sensitive to the modification that are introduced to the individuals cells during the construction of a mini-cube. In our scheme, we sum up the absolute values of candidate balance cells and select those with maximal sum value.

Note that in watermark insertion, a small portion of the cells, the bit positions of the selected cells, and the bit values assigned to the selected bit positions are all algorithmically determined under the control of a private key. The bit pattern constitutes the watermark. Without knowing the private key, an attacker is not able to know where exactly the watermark is embedded. We also note that the same HMAC function is used to determine the cells, the bit positions and the bit values in our scheme. To further increase the randomness of this process, different HMAC functions can be employed instead of single HMAC function.

3.2 Watermark Detection

The watermark detection algorithm is blind. It neither requires the knowledge of the original data cube nor the watermark in detection. Since the mini-cubes do not interfere

Algorithm 3. Watermark detection

```

1: for  $i = 1$  to  $N_x$ ,  $j = 1$  to  $N_y$ ,  $k = 1$  to  $N_z$  do
2:    $\mathcal{HMAC} = \mathcal{H}(\mathcal{K} \oplus opad, \mathcal{H}(\mathcal{K} \oplus ipad, X_i \circ Y_j \circ Z_k))$ 
3:   if  $(\mathcal{HMAC} \bmod \gamma \text{ equals } 0)$  then // This cell was marked
4:      $bp = \mathcal{HMAC} \bmod \xi$  //  $bp^{th}$  bit was marked
5:     totalcount=totalcount+1
6:     matchcount=matchcount+match( $\mathcal{HMAC}$ ,  $b(X_i, Y_j, Z_k)$ ,  $bp$ )
7:   end if
8: end for
9:
10:  $\tau = \text{threshold}(\text{totalcount}, \alpha)$  // See section 4.1
11: if  $(\text{matchcount} \geq \tau)$  then
12:   suspect piracy
13: end if
14:
15: match(MAC value  $\mathcal{HMAC}$ , cell value  $v$ , bit-index  $j$ ) return int
16: if  $(\mathcal{HMAC}$  is even) then
17:   return 1 if the  $j^{th}$  least significant bit of  $v$  is 0 else return 0
18: else
19:   return 1 if the  $j^{th}$  least significant bit of  $v$  is 1 else return 0
20: end if

```

with any cells that have been selected in watermark insertion, there is no need to consider the mini-cubes in watermark detection.

The watermark detection is shown in Algorithm 3. Line 3 determines whether a cell has been watermarked. Line 4 determines the bit position that have been watermarked. The function *match* compares the observed bit value at the bit position with the correct watermark bit that should be allocated at the position if the data is correctly watermarked. To claim the ownership over the detected data, one must know how many cells were tested (total count) and how many of them contained the expected watermark bit values (match count). In a probabilistic framework, only if a certain minimum number τ of cells contain the expected bit values, the ownership is claimed (see Line 10). The match count is compared with the minimum number, τ , which is returned by a threshold function (see Line 11).

A significance level α is used in the threshold function to determine the minimum match count τ (see Line 10). The significance level is the upper bound of the probability that the ownership is falsely claimed for a non-watermarked data cube. If the significance level is $\alpha = 10^{-9}$, for example, the probability of falsely detecting a watermark is less than 10^{-9} . A formal analysis on the detection probability and the threshold function is given in Section 4.1.

3.3 Extensions

The computational cost of our scheme is mainly determined by the amount of HMAC computation in the watermark insertion and detection. In our original scheme, an HMAC value is computed for each cell. The computation cost is $O(N_x \times N_y \times N_z)$ in terms of HMAC operations, where N_x , N_y , and N_z are the sizes of the feature attributes ($N_x \times N_y \times N_z$ is the total number of cells). We can extend the scheme such that an HMAC value is computed for each feature attribute value, rather than each cell. The HMAC value decides whether the feature attribute value is selected for embedding a watermark. An attribute value is selected if its HMAC modular $\sqrt[3]{\gamma}$ yields zero. On average, one

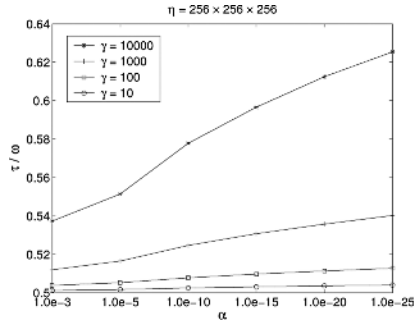


Fig. 3. Proportion of correctly detected marks required for claiming ownership in watermark detection with a significance level α

out of every $\sqrt[3]{\gamma}$ attribute values is selected in each dimension. A cell is selected for embedding a watermark if all its feature attributes are selected. Overall, one out of every $\sqrt[3]{\gamma} \times \sqrt[3]{\gamma} \times \sqrt[3]{\gamma} = \gamma$ cells is selected. This number is again the same as in our original scheme. For each selected cell, a bit position and a bit value can be determined based on the sum of the HMAC values that have been computed for the cell’s feature attributes. It is easy to know that the computational cost of this extended scheme is $O(N_x + N_y + N_z)$ in terms of HMAC operations. Compared with the original scheme, the extended scheme is more efficient for watermarking large data cubes.

4 Analysis

In the above section, we have designed a watermarking scheme for numeric data cubes and the scheme satisfies some particular requirements on aggregation, localization, non-watermarkable values, and computational cost. We now analyze the properties of the proposed scheme.

4.1 Detectability

To make our watermark detection reliable, the probability of falsely detecting a watermark from non-watermarked data must be low. This probability is controlled in our scheme by adjusting the significance level α and the number ω of watermarked cells. Bernoulli trials are employed for analyzing the probability of detecting each watermark bit from non-watermarked data. Since each watermark bit is determined by a HMAC pseudo-randomly, it has a probability of 1/2 to match the corresponding bit in non-watermarked data. Let $b(i; n, p)$ be the probability that n Bernoulli trials result in i successes and $n - i$ failures, where p is the probability of success and $q = 1 - p$ the probability of failure. The probability of having at least k successes in n trials (i.e., the cumulative binomial probability) can be written as

$$B(k; n, p) = \sum_{i=k}^n b(i; n, p) = \sum_{i=k}^n \frac{n!}{i!(n-i)!} p^i q^{n-i} \tag{1}$$

We now specify the threshold function that is used in Line 10 of Algorithm 3. Supposing $totalcount = \omega$, the watermark detection would examine ω “watermark bits”

from non-watermarked data. The probability of falsely detecting a watermark (i.e., at least τ out of ω “watermark bits” match their expected values by sheer chance) is $B(\tau; \omega, 1/2)$. Given the significance level α , the threshold function $threshold(\omega, \alpha)$ returns the minimum value τ such that $B(\tau; \omega, 1/2) < \alpha$.

The significance level α determines how amenable the watermarking system is to the false detection of a watermark. By choosing a lower value of α , the confidence level in the detection can be raised up if the detection algorithm discovers the owner’s watermark in a suspicious data cube. Figure 3 plots the required proportion (i.e., $\tau/\omega = \tau\gamma/\eta$) of the correctly detected marks for different values of α and γ . Clearly, we need to proportionately increase the number of the correctly detected marks as the value of α decreases. The figure also shows that the required proportion of the correctly detected marks decrease as the percentage of the watermarked cells increases. This illustrates that for larger data cubes, a smaller percentage of the total number of cells can be watermarked without increasing the significance level α .

4.2 Robustness

We now analyze the robustness of our watermarking technique against several malicious attacks including value modification, value selection, additive attack, and invertibility attack.

Value Modification. In a value modification attack, an attacker tries to destroy the owner’s watermark by modifying some cell values. There are various forms of value modification attack including bit-flipping, zero-out, and randomization. In all value modification attacks, the exact positions where a watermark is embedded are hidden from an attacker who does not know the private key. What an attacker can do is to apply the attacks to all cell values (or bits), or a portion of them in a random manner. The effect of these attacks is reflected in the portion V_m of the watermarked bits that have been changed/flipped. For example, if 30% of the least significant bits of all cell values are flipped in a bit-flipping attack, then the portion V_m is 30%. In a zero-out attack, we have $V_m = \min(\nu/(2\xi), 1/2)$ if ν least significant bits of each cell value are set to zero. In a randomization attack, one has $V_m = 1/(2r)$ if one out of every r cells is modified in a random manner. Given the portion V_m of the watermarked bits that have been changed, the sufficient and necessary condition that the embedded watermark can still be detected with a significance level α is $V_m \leq 1 - \tau/\omega$, or equivalently, $V_m \leq 1 - \tau\gamma/\eta$. Taking Figure 3 as an example, the portion V_m can be as large as $1 - 52\% = 48\%$ for $\gamma = 1000$, $\alpha = 10^{-5}$ and $\eta = 256 \times 256 \times 256$.

Value Selection. In a value selection attack, a subset of cells are selected from the original data cube with an intention that the embedded watermark cannot be detected from this subset of cells with a high probability. Typical value selection attacks in a data cube include data cube slicing, iceberg-cube, and random value selection. For all types of the value selection attack, the number ω' of the watermarked cells that are included in the value selection is proportional (i.e., $1/\gamma$) to the total number η' of the watermarkable cells that are included in this selection. By a value selection attack only, no errors are introduced to the watermarked values. Therefore, the watermark detection

will claim ownership for any non-zero significance level as long as ω' is greater than zero. The probability of $\omega' = 0$ is $(1 - 1/\gamma)^{\eta'}$, which can be made extremely low for a reasonably large η . For example, let $\eta = 256 \times 256 \times 256$ and assume that only one thousandth of the cells are selected in a value selection attack (i.e., $\eta' = \eta/1000$). Then the probability that no watermarked cells are included in the selection is about 5.9×10^{-74} for $\gamma = 100$, and about 5.1×10^{-8} for $\gamma = 1000$.

Additive Attack. In an additive attack, an attacker simply inserts his watermark into a data cube that has already been watermarked by its owner. If both watermarks are detected, then the ownership of the data is in dispute. To thwart this type of attack, Agrawal et al. [1] suggested to locate the overlapping regions of the two watermarks in which the bit values conflict. However, this may not be always possible if there is a value modification attack. To reach a decision with certain significance level, a enough number of watermarked cells that collide must be detected. This may not be realistic since the probability that the bit values of two watermarks conflict in any given cell is $\frac{1}{2(\gamma\xi)^2}$, which is extremely low for reasonably large γ (e.g., $\gamma = 100$). To solve this problem, one may consider using public watermark schemes (i.e., asymmetric watermarking, see [13]) that involve public key primitives such as trusted registration authorities and verifiable certificates to resolve any ownership dispute.

Invertibility Attack. An invertibility attack has also been identified in [1] for falsely claiming ownership using counterfeit watermarks. This attack discovers a key, which may or may not be the same as the original private key, to detect a satisfactory watermark from watermarked data for certain significance level α . This attack can be thwarted by imposing two additional requirements on watermarking schemes by convention. First, the private key should be long enough to thwart brute force search. Second, the significance level α should be low enough (e.g., 10^{-10}) such that the probability of accidentally finding a key that yields a satisfactory watermark is negligible.

5 Real Data Experiments

We now report experimental results that complement the analysis presented in Section 4. The experiments are performed using a real 3-D data cube, the LCDM (Lambda-dominated Cold Dark Matter) cluster simulations in astronomy, available from the Department of Astronomy and Astrophysics at the University of Chicago (<http://astro.uchicago.edu/daisuke/Research/simdata.html#threed>).

The data cube has 16,777,216 cells, and each cell value can be converted to an integer of 32 bits. In experiments, we vary γ from 1 to 10000, and fix ξ , λ_x , λ_y , and λ_z to 8. Our watermarking scheme is implemented in Visual C++ Version 6 using HMAC-SHA1¹ as the HMAC function. All the experiments are run on a HP Compaq computer with a Pentium(R) 4 CPU of clock rate 3.00GHz, 1.0 GB of RAM, and 40 GB hard-disk running Microsoft Windows XP.

¹ SHA-1 was recently found not as secure as it was believed to be [22]. Any one-way hash function can be used in our scheme. The reason for using SHA1 in our experiment is simply because of the availability of the HMAC-SHA1 code.

Table 2. Computational cost of watermark insertion and detection

	$\gamma = 1$	$\gamma = 10$	$\gamma = 100$	$\gamma = 1000$	$\gamma = 10000$
Number of cells selected for watermarking	16,777,216	1,676,954	167,124	16,764	1,703
Number of cells modified for watermarking	8,387,424	803,837	80,185	8,094	825
Time for embedding a watermark without mini-cubes (sec.)	299	158	144	142	141
Time for embedding a watermark with mini-cubes (sec.)	340	221	160	154	153
Percentage of watermarked cells for which mini-cubes can be constructed	0	100%	100%	100%	100%
Time for detecting a watermark (sec.)	254	158	143	141	140

5.1 Computational Cost

The first set of experiments evaluate the computational cost of the watermark insertion and detection. The experimental result is summarized in Table 2. The performance of our algorithms is measured in elapsed time. For the watermark insertion, the computational cost for constructing mini-cubes can be assessed by comparing the running time with mini-cubes and without mini-cubes.

In the watermark insertion, the computational cost can be broken down into three components: 1) The computation of a HMAC value for each cell, determining whether the cell is selected for embedding a watermark; 2) The modular computation for each cell that is selected, determining which bit position is selected, what the watermark bit value is, and whether the original cell value is to be modified for embedding the watermark bit; and 3) The computation of modifying a bit value and constructing a mini-cube for each cell that is modified.

The experimental results show that the computational cost increases as the percentage of the watermarked cells increases. In the extreme case where $\gamma = 1$, the number of the watermarked cells reaches its maximum. In this case, however, no mini-cube can be constructed, for every cell is selected for watermarking. Nonetheless, it takes longer for embedding a watermark with mini-cubes than without mini-cubes because of the time taken to attempt to construct mini-cubes. When γ is set to 10, the time required for mini-cube construction is $221 - 158 = 63$ seconds. When γ increases to 10000, the time used for mini-cube construction is down to 12 seconds because much less mini-cubes need to be constructed. In all of the experiments, the computation of HMAC values is the major component of the cost for watermark insertion.

Table 2 also gives the running time for watermark detection. The cost of watermark detection is similar to that of watermark insertion except that there is no need of mini-cube construction. Again, the major component of the cost is the computation of HMAC values. Note that the watermark insertion or detection can be done within five minutes in our experiments for a data cube with 256^3 cells. This result indicates that our algorithms have adequate performance to allow for their use in real world applications.

5.2 Imperceptibility

Since a data cube is a generalization of aggregation functions, the main service provided by a data cube (e.g., in OLAP) is to answer aggregation queries at various aggregation levels. To maintain the usefulness of data, the embedded watermark must be imperceptible; that is, it does not introduce intolerable errors to any aggregation queries that can be answered by a data cube. Consider the data cube shown in Figure 1. An example of

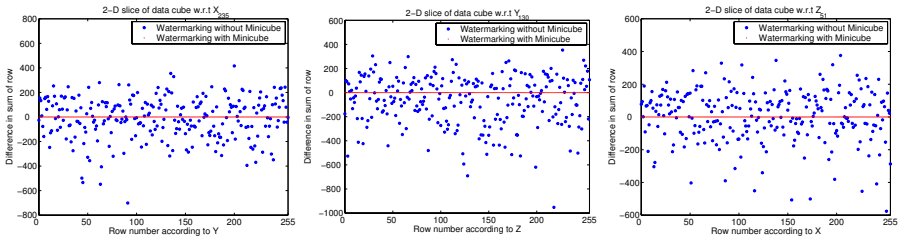


Fig. 4. Comparison of imperceptibility between two watermarking schemes

the aggregation queries asks for the total sale of red Ford GT cars. To answer this query, a row is localized in the data cube according to the *color* attribute “red” and the *make* attribute “Ford GT.” Then, all the cell values in the row are summed up to get the whole sale. To measure the effect to the aggregation queries of the watermark insertion, we calculate the differences between the sums of the original cell values and those of the watermarked cell values.

In our real data experiments, we evaluate typical aggregation queries that sum the cell values in each row in a randomly chosen slice, where a slice is a 2-D plane in the data cube. Let X, Y and Z denote the three feature attributes of the data cube, each of which has 256 values (i.e., from X_0 to X_{255}). A 2-D slice w.r.t. X_{235} , for example, denotes the 2-D plane in which the cells have the same attribute value $X = X_{235}$. For watermarking the data cube, we use $\gamma = 9$ and $\xi = 8$ in our experiments.

Figure 4 illustrates the differences in the sums of rows for three randomly chosen slices w.r.t. X_{235}, Y_{130} , and Z_{51} respectively. The differences in the sums are measured between the original and watermarked cell values. The differences indicate the errors that are introduced by the watermark insertion to the sum queries. We compare two types of watermark insertion, with mini-cubes and without mini-cubes. In Figure 4, the scattered points ‘*’ indicate the errors introduced by watermark insertion without mini-cubes. The points ‘.’, which are lined-up horizontally at zero value, indicate zero error that watermark insertion with mini-cubes always introduces. In comparison, the errors that are introduced by the watermark insertion without mini-cubes are notably large.

6 Conclusion

With the wide spread applications of data cube models in on-line analytical processing, security techniques for data cube ownership protection is becoming increasingly important. However, to our knowledge, little research work has been done to assert rights over distributed or sold data cubes.

In this paper, we proposed the first robust watermarking scheme for protecting the ownership of numerical data cubes. In our watermarking scheme, a data cube owner uses his private key to control watermark embedding parameters, including cell positions, bit positions, and specific bit values. Our blind detection algorithm requires neither the original data cube nor the watermark during watermark detection. The most prevalent data cube operations are aggregation queries. To eliminate errors introduced by watermark to aggregation queries, we invented a novel concept called mini-cubes.

Based on clearly defined optimization rules, a mini-cube is constructed for each cell that is modified in watermarking such that all sum queries in the watermarked data cube can be answered error-free, while without introducing any degradation on the robustness of the embedded watermark. In addition, we presented an extension of our basic scheme to improve watermarking efficiency when dealing with very large data cubes. We conducted extensive analysis as well as empirical evaluation for the proposed schemes in terms of watermark detectability, robustness, imperceptibility, and efficiency. Our results indicate that the schemes perform extremely well in real world applications.

Our future research efforts include development of public watermark (in which ownership can be publicly proved) and fragile watermark (by which value modification can be detected and localized) schemes for data cubes.

Acknowledgement

This paper is funded by Office of Research, Singapore Management University.

References

1. R. Agrawal, P. J. Haas, and J. Kiernan. Watermarking relational data: framework, algorithms and analysis. *The VLDB Journal*, 12(2):157–169, 2003.
2. R. Agrawal and J. Kiernan. Watermarking relational databases. In *Proceedings of the 28th VLDB Conference*, pages 155–166, 2002.
3. M. Atallah. A survey of watermarking techniques for non-media digital objects (invited talk). In *ACSW Frontiers 2005*, page 73, 2005.
4. M. Atallah, V. Raskin, C. Hempelmann, M. Karahan, R. Sion, U. Topkara, and K. Triezenberg. Natural language watermarking and tamperproofing. In *Fifth International Information Hiding Workshop*, pages 196–212, 2002.
5. M. Atallah and S. Wagstaff. Watermarking with quadratic residues. In *Proceedings of IS&T/SPIE Conference on Security and Watermarking of Multimedia Contents*, volume 3657, pages 283–288, January 1999.
6. M. Bellare, R. Canetti, and H. Krawczyk. Keyed hash functions and message authentication. In *Proceedings of Crypto'96*, pages 1–15, 1996.
7. E. Bertino, B. Ooi, Y. Yang, and R. Deng. Privacy and ownership preserving of outsourced medical data. In *Proceedings of the 21st International Conference on Data Engineering*, pages 521–532, 2005.
8. C. Collberg and C. Thomborson. Software watermarking: Models and dynamic embeddings. In *Proceedings of the 26th ACM SIGPLAN-SIGACT on Principles of Programming Languages*, pages 311–324, 1999.
9. I. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997.
10. I. Cox, M. Miller, and J. Bloom. *Digital Watermarking: Principles and Practice*. Morgan Kaufmann, 2001.
11. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.
12. D. Goss-Amblard. Query-preserving watermarking of relational databases and xml documents. In *Proceedings of the 19th ACM Symposium on Principles of Database Systems (PODS)*, pages 191–201, 2003.

13. G. Hachez and J. Quisquater. Which directions for asymmetric watermarking. In *Proceedings of XI European Signal Processing Conference (EUSIPCO), Vol. I*, pages 283–286, 2002.
14. N. F. Johnson, Z. Duric, and S. Jajodia. *Information Hiding: Steganography and Watermarking - Attacks and Countermeasures*. Kluwer Academic, 2000.
15. S. Katzenbeisser and F. Petitcolas. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, January 2000.
16. H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. In *Internet RFC 2104*, February 1997.
17. Y. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependable and Secure Computing*, 2(1):34–45, 2005.
18. R. Safavi-Naini. Tracing traitors: a selective survey. In *Digital Rights Management Workshop*, page 72, 2004.
19. R. Sion. Resilient rights protection for sensor streams. In *Proceedings of the Very Large Databases Conference*, pages 732–743, 2004.
20. R. Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. *IEEE Transactions on Knowledge and Data Engineering*, 16(12):1509–1525, 2004.
21. R. Sion, M. Atallah, and S. Prabhakar. Rights protection for categorical data. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):912–926, 2005.
22. X. Wang, Y. Yin, and H. Yu. Finding collisions in the full sha-1. In *The 25th Annual International Cryptology Conference*, Aug. 2005. <http://www.infosec.sdu.edu.cn/paper/sha1-crypto-auth-new-2-yao.pdf>.

An Efficient Probabilistic Packet Marking Scheme (NOD-PPM)

Huifang Yin and Jun Li

Computer Network Information Center,
Chinese Academy of Sciences, China
yinhuiifang@cstnet.cn

Abstract. This paper describes an efficient scheme of probabilistic packet marking. The main idea is to preserve the victims' IP addresses at the routers participating in the packet marking scheme, based on the precondition that a router won't begin to marking until it receives a signal from the victim. Then, the destination address field of IP header can be used to carry edge information without fragmenting, and the identification field can be used to check attack paths' validity under DDoS. We describe the scheme and discuss the number of packets required for reconstructing the attack paths, the number of false positives of attackers and the extra cost at routers in this paper.

1 Introduction

DoS(denial-of-service) attack floods the target host with a large amount of traffic, reducing the host's ability to service to legitimate users. In case of many attackers towards one target, it is called Distributed DoS (DDoS). 32% of respondents detected DoS attacks directed against their sites by 1999[6]. More severely, with the tools such as TBN, TFN2K and Trino, DDos can be easily launched.

IP traceback is considered to be a solution against DoS attacks, which aims at identifying the attack source. There are many ways of IP traceback such as link testing, logging, ICMP-based traceback and packet marking. Our approach is one kind of edge sampling marking belonging to Probabilistic Packet Marking (PPM).

Edge sampling algorithm was originally proposed by Savage et al [2]. They proposed an approach putting the routing information into the IP headers of packets. The victims can reconstruction the attack paths without the support from ISPs.

After [2], more attention is paid to PPM. Song and Perrig [3] improved the performance of PPM by assuming the victim possesses a complete network map. Instead of storing the address, this approach reduce storage requirement by storing a hash of each address. [3] also proposed an authenticated marking scheme to prevent compromised routers from forging or tampering markings from other uncompromised routers.

Drew Dean et al [4] proposed an algebraic approach using techniques from the coding theory and learning theory fields. Michael Goodrich [5] used large checksums to serve both as associative addresses and integrity verifiers.

In this paper, we propose a scheme maintaining victims' addresses at routers and using the destination address field of IP header to carry edge information. At the same time, the identification field is used to check the attack paths' validity in order to reduce the number of false positives of attackers under DDoS. We will refer our scheme as NOD-PPM (kNow Of Destination PPM) and the Compressed Edge Fragment Sampling using identification field of IP header in [2] as CEFS in the rest of the paper.

The rest of the paper is organized as follows: Section 2 introduces the CEFS in summary and then proposes the NOD-PPM scheme. Section 3 analyzes the NOD-PPM in several aspects, and experiments are carried out to show the performance of the NOD-PPM scheme. Section 4 comes the conclusion.

2 NOD-PPM Scheme

2.1 CEFS (Compressed Edge Fragment Sampling)

NOD-PPM is based on edge sampling [2]. CEFS is an interesting edge sampling scheme. In CEFS, each router marks its address information into packets with a probability p . In the case of flood-type DoS attack, the victim will collect enough marked packets and reconstruct the attack path. [2] uses the 16-bit identification field (seldom used) to store mark information: 5 bits for distance, 3 bits for index and 8 bits for fragments of the edge and its hash value. For more details, please refer to [2].

CEFS is interesting and elegant, but it has the disadvantage of the extraordinarily time-consuming process of grouping the fragments and the high false positive of attackers under DDoS attacks.

2.2 NOD-PPM (kNow of Destination PPM)

Fig.1 shows a network seen from the victim, V . R_i represent routers and A_i represent potential attackers. We call the tree with V as its root and A_i as its leaves *attack tree*, and call the paths from the root A_i to V *attack paths*.

When a victim site V suffers a Dos (or DDos) attack, it will send *marking-request-signals* to a set of routers requesting their participation in the probabilistic packet marking process. The set of routers may be in the same AS or under control of the same ISP or within d hops away from V , depending on the network's situation and the concern of the victim.

Each *marking-request-signal* contains the IP address of the victim (V) and its hash value $H(V)$. When a router receives a *marking-request-signal* and decides to take part in the edge marking process, it will insert an entry $(H(V), V)$ into the table T_{vic} (a table preserving victims' addresses at routers) and process the *Marking-and-Routing* procedure. If a router does not have a T_{vic} table or

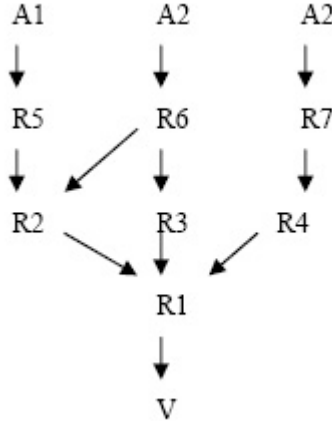


Fig. 1. Upstream routers map from a victim

its Tvic table is empty, it means that the router has not participated in any NOD-PPM process.

Each entry of Tvic will be deleted automatically a period of time after its creation, as the router supposes the victim has already have enough marked packets to reconstruction the attack tree. We assume that there are only a few victims in network that want routers to mark packets towards them at the same time. However, as the time of collecting enough packets for reconstructing attack tree is short (we will discuss it later), the routers can update the table Tvic quickly and server more victims. Besides the destination field of IP header, we use the undefined bit in flags field of IP header to denote whether the packet is marked, and use the identification field (only 0.25% packets use this field in the Internet [10]) to store distance, the hash value of the victim’s IP address and the digest of the router’s address (Fig.2) . More details are listed as follows:

(Let w be the packets marked.)

Destination address (w.dstIP, 32 bits): XOR of two adjacent routers’ addresses. If the marking router R is connected directly to V , this field is the address of R .

Undefined flag (w.marked, 1 bit): 1, where w is marked; 0, otherwise.

Identification 0-4 (w.distance, 5 bits): the distance from R_i to V , maximum 31, which is sufficient for almost all Internet Paths [7, 8, 9].

Identification 5-7 (w.IndVic, 3 bits): the hash value of victim’s IP address. For marked packets, routers refer to the Tvic table with it and find the victim’s IP address.

Identification 8-15 (w.dgstR, 8 bits): the digest of R_{i-1} (Suppose $\langle R_{i-1}, R_i \rangle$ is the marking edge, and R_{i-1} is closer to V). When there are several routers at the distance of i from V , we need to know whose address $w.dstIP$ should XOR with. This field will give us some information. If some R_{i-1} ’s digest is equal to $w.dstR$, we get R_i by $w.dstIP$ XOR R_{i-1} .

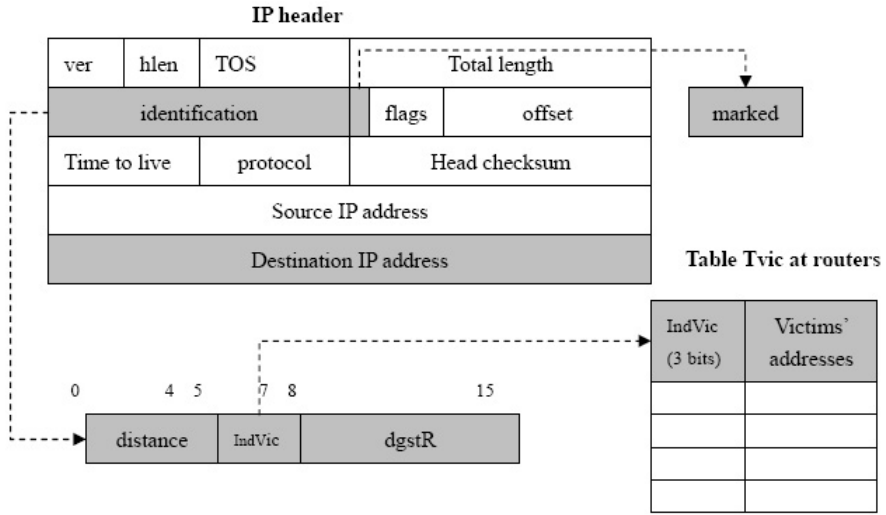


Fig. 2. Marking fields of IP header

When reconstructing, the victim firstly orders the marked packets by distance, then calculates the routers' IP addresses as Fig.3 shows. The algorithm is as follows.

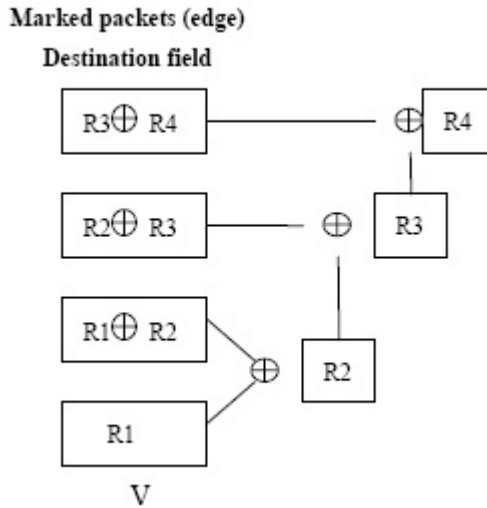


Fig. 3. Reconstructing the routers' addresses

Marking and routing procedure:(at router R)

```

for each packet w
{ let x be a random number from [0..1]
  if x<p //decide to mark
  {
    w.IndVic = H(w.dstIP)
    w.dstIP = R
    w.marked = 1
    w.distance = 0
  }else // not to mark
  {
    if w.marked==1 //marked packet
    {
      if w.distance==0
      {
        w.dstIP = w.dstIP XOR R
        w.dgstR = Dgst(R)
      }
      w.distance++
      dstadd = Tvic(w.IndVic)
      route w to dstadd
    }else //unmarked packet
      do regular routing
  }
}

```

Reconstruction procedure:(at victim V)

```

order marked packets by the value of w.distance;
let V be the root of the attack tree T;
for all marked packets with w.distance==0;
{
  add the edge(v,w.dstIP) to T;
  calculate the digest for every leave l;
}
for distance = 1 : max_distance
{
  if w.distance==l.distance && w.dgstR==Digest(l)
    add edge(w.dstIP XOR l , l) to T
  calculate the digest for every leave l
}

```

3 Analysis

3.1 Packets Required for Victims to Reconstruct Attack Paths

It is obvious that the probability of receiving a marked packet form a router d hops away from the victim is $p(1-p)^{d-1}$. The victim always collects more packets marked by near routers, so the number of packets needed for reconstruction (N) depends on the arrival of the packets marked by the routers closest to the attackers. As the [2] analyzes, N follows the bounded expectation: (d is the distance between A_i and V .)

$$E(N) < \frac{\ln(d)}{p(1-p)^{d-1}}, \text{ where single attack path, no fragmenting.}$$

$$E(N) < \frac{k\ln(kd)}{p(1-p)^{d-1}}, \text{ where single attack path, } k \text{ fragments.}$$

Similarly, we can get:

$$E(N) < \frac{m\ln(md)}{p(1-p)^{d-1}}, \text{ where } m \text{ attack paths, no fragmenting.}$$

We can see our scheme requires less than 1/8 times of the packets CEFS requires. For example, let $d=10$, $m=1$, and $p=1/25$, the CEFS needs about 1300 packets on average to perform reconstructing process [2], while NOD-PPM requires less than 90 packets. In other words, it cost routers less time to mark for each victim. Therefore, though the T_{vic} table can preserve only a few victims' addresses, the table can update quickly for the new coming *marking-request-signals*. In DDos attacks, $E(N)$ increases with the number of attackers. For example, if there are 100 attackers and the average packet arrival rate is 100p/s, it will cost 250s which is still reasonable.

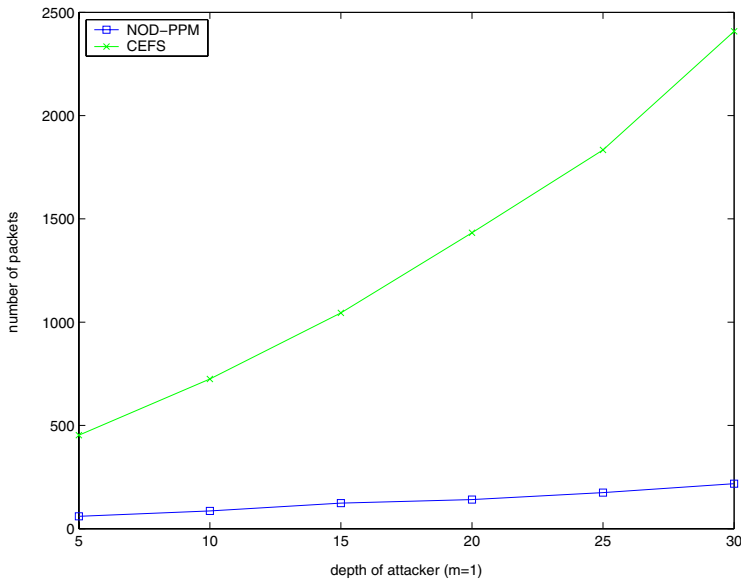


Fig. 4. Number of packets ($m=1, p=1/25$)

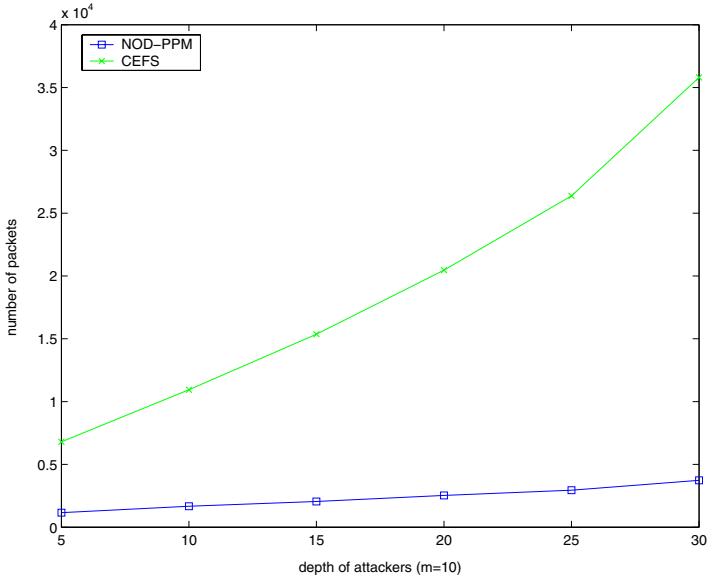


Fig. 5. Number of packets ($m=10, p=1/25$)

Fig.4 and Fig.5 shows the experimental results of the number of packets needed for reconstruction. In the experimental we choose $p=1/25$ and assume the attack(s) generate(s) packets at a constant rate. ([9] do several experiments on the time needed to collect enough packets in different traffic arrival processes, including Poisson arrivals, constant-rate arrivals, and burst mode arrivals. The results seem close to each other in different arrival processes.) Fig.4 shows the case with only one attacker ($m=1$), while Fig.5 shows the case with 10 attackers ($m=10$). The y-axis "number of packets" in both figures is the average of more than 100 tests. We also assume that all the attackers are in the same depth in Fig.5 for simpleness. We can see that the number of packets needed grows significantly more slowly in the NOD-PPM than in CEFS with the growing of the depth of attackers.

3.2 Reconstructing Time

If there is only one attacker, NOD-PPM performs almost the same as the CEFS does. However, under DDos, grouping the edge fragments is extraordinarily time-consuming. If there are m completely independent edges at the depth of i , there will be $O(m^8)$ combinations to be considered just at the depth of i . Let $m=10$, it is 100 million. For every combination, the hash value of the edge must be calculated and compared. Since a victim always has limited computing resource, it is hard to execute the reconstructions in a reasonable time.

On the other hand, in NOD-PPM, edge information is not fragmented by using the 32-bit destination address field, so the process of grouping is totally omitted.

3.3 The Number of False Positives of Attack Paths (NPF)

Another benefit of NOD-PPM comes from the 8-bit **dgstR** field. Under DDos attacks, there are several routers at depth of i ($i=1,2,d$) in the attack tree. When calculating the routers at the depth of $i+1$ of the tree, the victim does not know which router should be chosen as downstream router without **dgstR** fields. The NPF will increase exponentially, in the worst case, $O(m^d)$. With **dgstR**, if the digest function is ideal, the complexity of the problem can be reduced $\min(28, m)$ times at every depth. (m is the number of attackers and d is the maximum depth of the attack tree.)

To test the benefit, we simulate the processing of reconstruction with Internet topology dataset obtained from Lucent Bell Lab [1]. (As our concern is the collision of IP addresses' hash values, we leave the routers with "HOLE_" identifier out.) We randomly select attackers 10 hops away from the victim. The result is in Fig. 6.

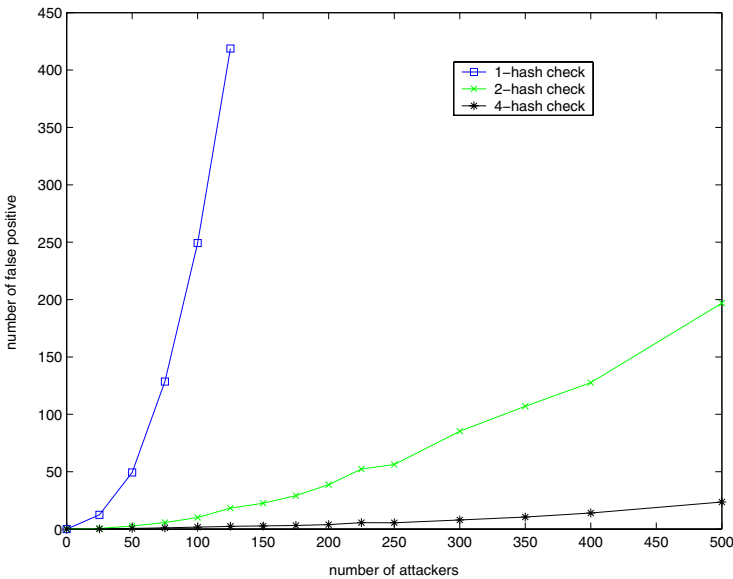


Fig. 6. False positives of NOD-PPM

Though using digest of router's address reduces false positives, the performance is not good enough. 8-bit digest is not sufficient for 32-bit address. To improve the performance, we try to use two or more independent hash functions. In this case, we should modify our algorithm. When marking, the router choose one of these hash functions in some order to generate the **dgstR** field; when reconstructing, we consider one edge to be linked to a leaf router in attack tree only when all of the router's hash values are matched by some packets with the same **dstIP** filed. We can use part of the **IndVic** field (1 bit for 2 hash functions, 2 bits for 4 hash functions) to denote the chosen function in packets.

In Fig.6, we can see there are only about 25 false positives attacked by 500 attackers in 4-hash check scheme. The performance result is good, though this modification causes the increase of the number of packets needed for reconstruction and decrease the number of entries in T_{vic} .

3.4 Cost at Routers

Obviously, the NOD-PPM increases the routers' work. As long as a router participates in the NOD-PPM process, it will check every packet transiting it to make sure whether the packet is marked. In addition, a table T_{vic} is needed. However, since regular routing process will check the fields of IP header such as destination address, TTL, checksum and so on, checking one more field (the "marked" field) won't significantly increase the routers' load. Moreover, compared to routing table and forwarding table in a router, the table T_{vic} is tiny (at most $8 \times (32+3) = 35B$), so searching and reading T_{vic} is fast.

3.5 Limitations

The NOD-PPM is based on the request-marking-signal, but the signal may be under attacks. If the request-marking-signal is invalid, the whole packet-marking process is meaningless. Besides, the table T_{vic} at routers should be synchronized in some way.

If some routers may not take part in the NOD-PPM scheme, marked packets will be delivered incorrectly. If some routers are compromised, they will send out false information. Therefore the scheme depends on the routers' participation and reliability.

As all packet marking schemes, the scheme can only find the nearest slaver attackers and require enough packets from every attacker.

4 Conclusion

In this paper, we proposed a modified PPM approach to IP traceback called NOD-PPM. *Require-marking-signal* is prerequisite to our approach, which should be sent from the victim and received by the routers correctly. Looking up the table T_{vic} (preserving addresses of the sources of *Require-marking-signals*), the routers know exactly the destination of the marked packets, so the destination address field of IP header can be used to carry the edge information without fragmenting. Though the table T_{vic} is small, it updates quickly according to our discussion. This modification reduces the numbers of packets required to reconstruct attack paths by 1/8 times at least and eliminates the time-consuming processing of grouping fragments in DDos. In addition, the introduction of digests of routers' addresses can decrease the number of false positives of attackers in DDos. The NOD-PPM increases the routers' work, but it is little compared to the regular work of routers.

References

1. Internet mapping, <http://cm.bell-labs.com/who/ches/map/dbs/index.html>, 2006.
2. S. Savage, D. Wetherall, A. Karlin, and T. Anderson.: "Practical Network Support for IP Traceback," Proc. 2000 ACM SIGCOMM Conf., pp. 295-306, Aug. 2000.
3. D.X. Song and A. Perrig.: "Advanced and Authenticated Marking Schemes for IP Traceback," Proc. IEEE INFOCOM Conf., Apr. 2001.
4. D. Dean, M. Franklin, and A. Stubblefield.: "An Algebraic Approach to IP Traceback," Proc. Network and Distributed System Security Symp. (NDSS '01), Feb. 2001.
5. M. Goodrich.: "Efficient Packet Marking for Large-Scale IP Traceback," Proc. 9th ACM Conf. Computer and Communication Security, ACM Press, pp.117-126, 2002.
6. Computer Security Institute and Federal Bureau of Investigation. 1999 CSI/FBI Computer Crime and Security Survey. Computer Security Institute publication, Mar. 1999.
7. Robert Carter and Mark Crovella.: "Dynamic server selection using dynamic path characterization in wide-area networks," Proc. INFOCOM Conference, April 1997.
8. Wolfgang Theilmann and Kurt Rothenmel.: "Dynamic distance maps of the internet," Proc. IEEE INFOCOM Conference, March 2000.
9. Terence Law and John Lui.: "You Can Run, but You Can't Hide: An Effective Statistical Methodology to Trace Back DDos Attackers," IEEE Transactions on Parallel and Distributed System, Vol.16, No.10, October 2005.
10. I. Stoica and H.Zhang: "Providing Guaranteed Services Without Per Flow Management," Proc. ACM SIGCOMM Conference, Boston, MA, 1999.

Resistance Analysis to Intruders' Evasion of Detecting Intrusion

Jianhua Yang¹, Yongzhong Zhang², and Shou-Hsuan Stephen Huang¹

¹ Department of Computer Science, University of Houston
4800 Calhoun Rd. Houston, TX, 77204 USA
{jhyang, shuang}@cs.uh.edu

² Department of Computer Science, Shanghai TV University
288 Guoshun Rd, Shanghai, 200433 China
yzhang@shtvu.edu.cn

Abstract. Most network intruders launch their attacks through a chain of compromised hosts (stepping-stones) to reduce the risks of being detected or captured. Detecting such kind of attacks is important and difficult because of intruders' evasion to detection, such as time perturbation, and chaff perturbation. In this paper, we propose a clustering algorithm to detect stepping-stone intrusion based on TCP packet round-trip time to estimate the downstream length of an interactive terminal session and give its resistibility to intruders' evasion. The analysis and simulation results show that this algorithm can detect stepping-stone intrusion without false alarm, and low misdetection. It can resist to intruders' time perturbation completely, as well as chaff perturbation to a certain extent.

Keywords: Network security, intrusion detection, stepping-stone, evasion, time perturbation, chaff.

1 Introduction

Network Intruders tend to launch their attacks through compromised hosts, which are called stepping-stones [1], to reduce the risks to be detected or captured. Such kind of attack is called stepping-stone intrusion. To detect stepping-stone intrusion, bunch of approaches [2], [1], [3], [4], [5], [6], [7], [8] have been proposed since 1995 even though the concept 'Stepping-stone' was formally defined in 2000. Some approaches, such as the method in paper [2], are unavailable in detecting encrypted TCP interactive sessions because they are content-based. Some other approaches, such as [1], and [3], are vulnerable to intruders' time and chaff perturbation. Even though the approaches presented in [5], and [6] have the ability to detect intruders' evasion, as well as the theory analysis of detecting stepping-stone on jittered sessions, but they all suffer from high false alarm rate in detecting intrusion because being used as a stepping-stone does not mean an intrusion. The paper [1] pointed out that legitimate stepping-stone users could be as high as 100 each day in the one large site of the University of California at Berkeley. Our experiment also showed that the false alarm rate could be as high as 80% sometimes in the main site of Computer Science Department, University of Houston by using the approach in [1]. The main reason is

that many applications (users), which are obviously not intruders, need to connect out through one or two hosts to do something necessarily.

However, through long time observation and careful analysis, we found that such legitimate users (applications) connect out through at most two hosts, most of the time, only one host. As intruders, they prefer to compromise more hosts to make them safe, usually more than three hosts. This motivated us to detect stepping-stone intrusion by estimating the number of hosts compromised to reduce false alarm rate. K. H. Yung proposed an approach in paper [4] to estimate the length of an interactive session by exploiting the TCP packet round-trip time (RTT). Its main idea is to use the length of a connection from the monitor point (one of the stepping-stones) to the next neighbor host to measure the length of the whole session relatively. An intrusion is detected if the whole session is relatively long. This method can largely reduce the false alarm rate in detecting stepping-stone intrusion, but still suffer from 1) yard-stick problem [9]; 2) computing TCP packet RTTs of an interactive session inaccurately.

J. Yang and S. Huang proposed an algorithm, which can overcome yard-stick problem, to detect stepping-stone intrusion in papers [7], [8]. The basic idea of Yang's approach is to estimate the number of connections of an interactive session by monitoring all the TCP 'send' and 'echo' packets from the start to the end of the session, finding all the matches ('match' will be defined later) between each send and echo packet, and computing all the RTTs between matched send and echo packets. When a session has only one connection from the monitor point at the beginning, we get bunch of RTTs which are different, but should be bounded within a narrow range, called a level. Similarly, when the session has two, three, even more connections, we should get RTTs in different levels. In other words, if we compute all the RTTs, and count the number of levels, we are supposed to know the number of the connections of the session. The accuracy of this method depends on the performance of an algorithm to match send and echo packets. Two algorithms, the Conservative and the Greedy, are proposed in Yang's paper [8]. The Conservative algorithm could give high quality match but with only little packets matched; the Greedy algorithm could give more 'matched' packets but with some unsure matches. A novel algorithm called Clustering-Algorithm, which can get both high matching-accuracy and matching-rate, is proposed in this paper to detect stepping-stone intrusion.

In this paper, first we describe the Clustering-Algorithm, and then mainly focus on analyzing its resistibility to time and chaff perturbation. The analysis result shows that this algorithm is robust in detecting intruders' time perturbation, as well as robust to a certain degree in detecting intruders' chaff evasion. The experiment and simulation results also showed that the Clustering-Algorithm can detect stepping-stone intrusion with zero false alarm rate and very low misdetection. The miss detecting would be as low as zero if we assumed that the detecting point is always far away from the victim site. The detail analysis is in Section 2.

The rest of this paper is arranged as following. Section 2 gives the problem statement. Section 3 describes the Clustering-Algorithm and its experimental verification. Section 4 and 5 focuses on analyzing the resistibility of the Clustering-Algorithm to intruders' manipulation. In Section 6, related work is discussed. Finally, in Section 7, the whole work is summarized and future work is presented.

2 Problem Statement

A network intruder may compromise some hosts h_1, h_2, \dots, h_n to invade any host h_{n+1} , which is assumed the victim site, from host h_0 . Before launching his attack, the intruder may establish a TCP session through first connecting to h_1 from h_0 , then to h_2 from h_1 , and eventually to h_{n+1} from h_n by using tools, such as Telnet, SSH, rlogin, and so on. The hosts h_1, h_2, \dots, h_n are called stepping-stones on any of which we are able to run our detecting program. The host we can run our detecting program is called a monitor point (MP). We use flow $h_i \rightarrow h_{i+1}$ to represent a network connection, and the sequence of network connections $h_0 \rightarrow h_1 \rightarrow \dots \rightarrow h_i \rightarrow h_{i+1} \rightarrow \dots \rightarrow h_n \rightarrow h_{n+1}$ is called a connection chain which length is $n+1$ connections. The chain toward the victim side from a MP is called a downstream connection chain. Correspondingly, the chain toward the intruder side from the MP is called an upstream connection chain. In this case, if we assume h_i is a MP, and $0 < i < j < n+1$, then the chain $h_0 \rightarrow h_1 \rightarrow \dots \rightarrow h_i$ is called an upstream connection chain of the MP, and $h_i \rightarrow h_{i+1} \rightarrow \dots \rightarrow h_{n+1}$ is called a downstream connection chain. The connection $h_i \rightarrow h_{i+1}$ is an upstream connection of the connection $h_j \rightarrow h_{j+1}$, while $h_j \rightarrow h_{j+1}$ is a downstream connection of $h_i \rightarrow h_{i+1}$. The length of a downstream (upstream) connection chain is called a downstream (upstream) length.

For each MP, an incoming connection is a connection connected to the MP, and an outgoing connection is the one connected out from the MP. For example, if h_i is a MP, then $h_{i-1} \rightarrow h_i$ is its one incoming connection, and $h_i \rightarrow h_{i+1}$ is its one outgoing connection. Most of the previous approaches focus on detecting if a host is used as a stepping-stone by correlating incoming and outgoing connections; these methods introduce lots of false alarms in detecting stepping-stone intrusion. To reduce the false alarms, our approach to determine an intrusion is first to estimate the downstream length and then compare it with a given threshold (in this paper, we assume it is 3) to see if a user comprises more hosts. Obviously the misdetection depends on where the MP is in a connection chain. Actually we could not appoint which host is our MP because the reality is that we even do not know which host is going to be used as a stepping-stone. What we could do is to install our detecting program in all the hosts (host-based detection) or gateways (network-based detection) which are under control. In this case, if we assumed our MP is the host h_n , we would miss the possibility to detect any intrusion because the downstream length is only one connection. However, if we assumed our MP is the host h_1 , we would not miss any intrusion. The closer to the victim's host a MP, the higher misdetection the detecting algorithm. In this paper, to reduce misdetection, we simply assume that our MP is far away from the victim site. So the statement of detecting stepping-stone intrusion is reduced to estimating a downstream length.

Intuitively, all the packets in an incoming connection of a MP are forwarded to an outgoing connection if the two connections are in a same chain. For a TCP session, our algorithm only focuses on two kinds of packets: Send and Echo, which are defined as the following,

Send: A TCP packet is defined as a Send if it propagates downstream and has either both flags 'Push (P)' and 'Acknowledgement (A)' or only flag 'P' [10];

Echo: A TCP packet is defined as an Echo if it propagates upstream and has either both flags 'Push (P)' and 'Acknowledgement (A)' or flag only 'P'.

In each outgoing connection of a MP, we put the packets captured into two sequences S and E depending on if a packet is a Send or an Echo, respectively. If we monitor an outgoing connection for a certain period of time, we have $S=\{s_1, s_2, \dots, s_n\}$, and $E=\{e_1, e_2, \dots, e_m\}$ where we use s_i (e_j) to represent the timestamp of a Send (an Echo), $1 < i < n$ ($1 < j < m$). If a Send matches an Echo, then the gap between them is called the RTT of the Send; this gap can also be used to measure the downstream length of a connection chain at that time. For example, if (s_i, e_j) is a matching pair, the gap $e_j - s_i$ can be used to represent the downstream length. We give the definition of 'Match' as the following,

Match: If a given Echo is directly triggered by a Send, then the Echo is defined as a matched packet of the Send.

It would be trivial to match send and echo packets if only one Echo was triggered by only one Send. The complicated cases are more Sends correspond to one Echo or one Send corresponds to more Echoes. Under these situations, we use the shortest gap to measure the downstream length based on the definition of Match. For example, if send packets (s_1, s_2, \dots, s_k) only trigger one Echo (e_j), then the RTT between them is $e_j - s_k$; if one Send s_i triggers more Echoes (e_1, e_2, \dots, e_q), then the RTT between them is $e_1 - s_i$.

Through our observation, we found that the case more Sends trigger one Echo seldom occurred in an interactive session. Even though it happened occasionally, we also found that the timestamps among these Sends are so close that we might consider them as one Send in terms of computing the RTT. To simplify our analysis, we assume that different Sends will trigger different Echoes. Each Send has and only has one RTT. So if we assume that a connection chain does not increase or decrease in a certain period of time, for all the n send packets captured in the outgoing connection, we have n RTTs that are different but within a very narrow range, which is called one level. Here is a real world example which may give us a practical sense of RTTs. We take one host (Ac108) of our lab as a MP, and connect out to a host in California through five hosts spanning from the United States to Mexico. Part of the RTTs is (212.34, 216.02, 211.37, 212.63, 210.71, 216.55, 212.26, 211.52, 213.10, 212.34, 218.24, 215.57, 212.93, 215.82, 215.91, 211.61, 212.49) in *ms* (millisecond). We increased the chain one more connection, and got another RTT sample: {265.26, 265.47, 264.06, 262.91, 276.74, 263.55, 266.37, 266.75, 264.83, 265.32, 265.59, 265.16, 262,91} in *ms*. Look at these two real world RTTs examples, we find that even though the RTTs in each sample vary, but the RTTs in the first one is bounded within the range <210, 219>, and in the second one bounded within the range <262, 267>. It is not difficult to distinguish the two levels unless they are overlapped. In other words, we can estimate the number of connections of a chain by counting the number of RTTs' levels through monitoring a connection chain from the start to the end. Apparently, the key is to compute the RTTs or to find matching pairs between Sends and Echoes. Now the statement of estimating a downstream length in terms of connections is reduced to finding the RTTs of the Sends of a connection chain.

There is one issue we must mention, which is how to get the RTTs before we propose the Clustering-Algorithm and guarantee the RTTs are correct. One obvious way is to use Telnet to set up a chain and match the packets based on their contents. But here we use a simple way, and still efficient to generate the correct RTTs. In fact, the reason we cannot match each Send with its corresponding Echo or Echoes is that usually a user types so fast that there are many Send-Echo overlaps which make it

difficult to match a Send and an Echo. An example here could be helpful for understanding this point. As a regular user, his/her typical keystroke speed is about 6-10 keys per second, which means the gap between two consecutive keystrokes is about 100-166 *ms*. Here we assume each keystroke generates one packet. Suppose we take the upper bound 166 *ms*, it means the Echo corresponding to a Send cannot come back before the next Send is sent if the length of a connection chain is longer than 166 *ms*, such that the scenario $\{s_1, s_2, e_1, e_2\}$ would be generated. In this case, we are pretty sure that s_1 can match e_1 , but we cannot guarantee that s_2 can match e_2 because it is possible that s_1 could match both e_1 and e_2 , and s_2 is going to match the packet after e_2 . But if the situation turns into the case, like $\{s_1, e_1, s_2, e_2\}$, it should be clear that s_1 matches e_1 , and s_2 matches e_2 whatever what kind of packets follow e_2 . To generate this kind of situation, we only need to control our typing speed slow enough to guarantee that the Echo of one Send comes back before the next keystroke. So under this situation, if the packet sequence captured was like $\{s_1, e_1, s_2, e_2, e_3, s_3, e_4, \dots, s_n, e_m, e_{m+1}, \dots, e_{m+k}\}$, finding the RTTs of all Sends would be effortless.

3 Clustering-Algorithm

3.1 The Basic Idea

Given two sequences $S=\{s_1, s_2, \dots, s_n\}$ and $E=\{e_1, e_2, \dots, e_m\}$, we assume that these packets are captured in one outgoing connection of a MP from the time a user connects to the first downstream host to the time the user connects to the victim site. Our purpose is to find the RTTs for all Sends in S . If we knew the matched pairs between S and E , it would be simple to find the RTTs. However, matching each Send in S with an Echo in E is the same difficult as finding the RTTs. Through the discussion in Section 2, we know that the RTTs in each level are within a very narrow range. We monitored many connection chains their length are fixed for a long time, and got hundreds of thousands RTTs. We found the occurrence of the RTTs of each session obeys Poisson distribution. One of the distribution results is showed in Fig.1, where Y axis represents the occurrence probability of a specific range of RTT, and X axis represents the value of each range of RTT in unit microsecond.

Fig.1 shows clearly that more than 95% of the RTTs distribute around 137,000 in microsecond. It means if we mined all the RTTs, more than 95% RTTs would be in one cluster with center 137,000 in microsecond. Any Send s_i in S must be matched by one or more Echoes in E , but we do not know the one or ones exactly. However, we do know that the Echoes of E only after s_i have possibility to match s_i ; we assume those Echoes are $\{e_j, e_{j+1}, \dots, e_m\}$. By computing the gaps between s_i and each of e_j, \dots, e_m , we get a data set $S_i=\{s_i e_j, s_i e_{j+1}, \dots, s_i e_m\}$ in which any element $s_i e_v$ represents the time gap between s_i and e_v , that is $s_i e_v = e_v - s_i$ (v is an integer, and $j \leq v \leq m$). Even though we do not know which element in S_i is the right one to represent the RTT of s_i , but we do know it must be one element of S_i . If we only check one data set S_i , it might be difficult to determine an element in S_i to represent the real RTT. However, by checking more such kind data sets, we found a very interesting and useful point that is each data set contains one element which is very close to an element in other data sets. In other words, if we check data sets $S_i, S_{i+1}, \dots, S_{i+k}$, most probably there is one

element in S_i which is very close to one element in S_{i+1} , as well as close to an element in $S_{i+2}, S_{i+3}, \dots, S_{i+k}$, respectively. The more data sets we check, the higher probability we find the RTTs precisely. The above discussion is based on the length of a connection is fixed. If we monitor a connection chain which length is increasing from the start to the end, and capture all the n Sends, and m Echoes, we get n data sets S_1, S_2, \dots, S_n each of which corresponds to one different Send. We combine all the data sets together to form a set $D, D=S_1 \cup S_2 \cup \dots \cup S_n$, where \cup represents union operation. If we use data mining method to mine D , the number of the significant clusters (defined below) is the downstream length in terms of connections.

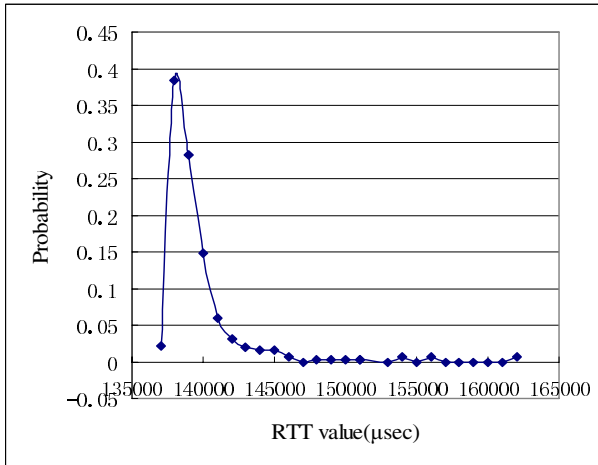


Fig. 1. The distribution of RTTs of an interactive session

3.2 Clustering-Algorithm

Based on the above basic idea, we propose a Clustering-Algorithm to find the RTTs of all the Sends in a connection chain by exploiting the distribution feature of RTTs.

Step 1 of this algorithm is simply a clustering algorithm, the maximum-minimum distance clustering (MMD), with a predefined threshold th . The MMD algorithm can result in v clusters. We shall assume the elements of these clusters are sorted in an increasing order of the index i which is the send packet index. The range of a cluster C is defined as the maximum i -index of the elements in C minus the minimum i -index plus one. In Step 2, we filter out duplicated elements either with the same send packet or with the same echo packet in each cluster. The preference is given to a smallest RTT.

In Step 3, we measure the likelihood of a cluster truly representing a level in the RTTs. A true RTT cluster should have elements representing consecutive send packets with very few exceptions. So we first define a subset of a cluster containing “Connected” elements, i. e., elements having neighbors with a distance of g . A typical value for g is 2 (allowing one missing send packet). “Disconnected” elements are mostly not part of this cluster.

The purpose of Steps 4 and 5 is to select the clusters that have very high likelihood of true RTTs. According to the Chebyshev inequality, there are very few elements in

the set R outside two standard deviations of the mean of the set R . A true RTT cluster is a partition of all send packets satisfying the following two conditions: (1) all clusters are mutually disjoint, and (2) the union of all clusters is equal to the whole send packet set. In reality, condition (1) is easy to be satisfied, but it is very difficult to make the union of the clusters exactly equal to the send packet set S . In our algorithm it turns out to find the clusters which union has the largest distribution in S .

Clustering-Algorithm (D, th , g):

1. Call $MMD(D, th)$, output clusters C_1, C_2, \dots, C_v .
2. For each cluster C , (1) if $t(i,j), t(i,k) \in C$, and $j < k$, delete $t(i,k)$, and (2) if $t(i,j), t(k,j) \in C$, and $i < k$, delete $t(i,j)$.
3. For each cluster C , compute the clustering ratio $r_i = |C_n|/range(C)$, where $C_n = \{t(i,j) \in C \mid \exists t(p,q) \forall C, \text{ and } |i-p| \leq g\}$.
4. Select clusters X_1, \dots, X_s from the clusters C_1, C_2, \dots, C_v . They have significantly higher ratios among set $R = \{r_1, \dots, r_v\}$. We consider r_i to be significantly higher than the rest of the values if it is two standard deviations higher than the mean of R ; the corresponding cluster is called significant cluster.
5. Find a maximum disjoint subset among X_1, \dots, X_s . If there is a tie, select the subset whose union is the largest.
6. Output the number of the clusters in Step 5 as the number of connections.

End

The detailed MMD algorithm can be found in [11], [12], [13]. It first finds all the cluster centers, and then adds all the elements to each cluster. Its cluster results depend on its first element, rather than the input elements order. If we assume $m=n$, the time complexity of MMD is $O(n^2v^2)$ with the worst case $O(n^4)$. The MMD algorithm still has many computations. One future work is to make MMD more efficient by reducing the space of set D .

3.3 Experimental Verification

Intuitively the above algorithm is correct, but so far we cannot prove its correctness in terms of estimating the downstream length of a session in theory. In this section we justify its correctness through experimental result. We did the experiments hundreds of times under different situations, each time it can give us the expected results. Here we show one of the results.

We set up a connection chain by using SSH starting from one host (Acl08) of our lab to a host VH in California which is assumed the victim's site, and the connection is $Acl08 \rightarrow Acl09 \rightarrow H_1 \rightarrow H_2 \rightarrow H_3 \rightarrow H_4 \rightarrow VH$, where a user operates at host Acl08 and we take Acl09 as the MP. H_1, H_2, H_3 , and H_4 are remote hosts spanning the United States and Mexico. We monitored the outgoing connection of Acl09, captured all the Sends and Echoes, and ran the Clustering-Algorithm with parameters $th=0.1$, and $g=2$. The final result is showed in Fig. 2 in which Y axis represents the clustering ratio, and X axis represents the cluster center value in ms .

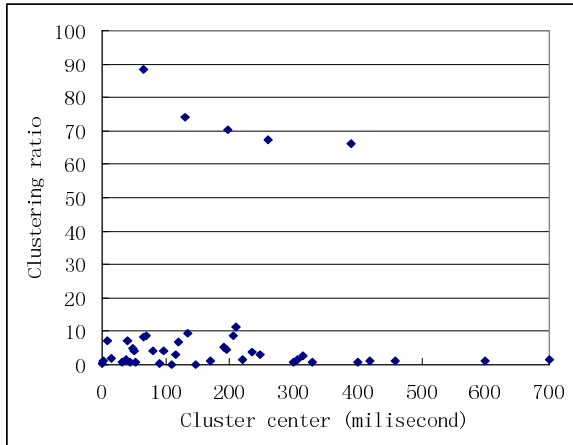


Fig. 2. One of the cluster results of the algorithm

In this experiment, we already knew that the connection chain has five connections from the MP. Even though there are more than 50 clusters got by MMD, but only five of them have the clustering ratios higher than 60% while the others are lower than 11%. The number of the clusters with significant ratios is exactly the same as the downstream length in connections. Fig.2 shows that the five significant clusters center on 65.7, 130.4, 197.3, 260.5, and 390.4 *ms*, which represent five levels of the RTTs, respectively. For example, the value 65.7 *ms* means most of the RTTs are around this value when the chain is at the level with one connection. As we have discussed before, the accuracy of the Clustering-Algorithm depends on the number of Sends captured. An intruder might evade the detection by typing characters as little as possible to make this detection unable to work. But this would be difficult because a user must type the username, password, and a domain name which are necessary information for connecting to another host. And more, once an intruder logs into one host, he/she must do something to make sure there is no surveillance program in this host before connecting out. To avoid typing too much would put an intruder in a very embarrassed environment and make it hard to invade other hosts. In fact, most intruders usually evade detection by conducting time and chaff perturbation. In the following two sections, we center our discussion on how the Clustering-Algorithm resists to intruders' evasion.

4 Resistance to Time Perturbation

We have proposed the Clustering-Algorithm to detect stepping-stone intrusion by estimating the downstream length of a connection chain. The experimental results showed that it works perfect without any perturbation on the session. How does it work under perturbation? In this section we center on analyzing the performance of the Clustering-Algorithm under time perturbation.

For an interactive session established by an intruder, it could be perturbed at any one or some of the hosts of a connection chain. To simplify our analysis, we assume

that an intruder could conduct time perturbation on any one host of a chain, rather than on some hosts. Like the assumption in paper [5], [6], we assume that the perturbation is bounded, as well as assuming that the perturbation can only be conducted on the Sends of an outgoing connection, rather than the Echoes of both outgoing and incoming connection. Even though an intruder could be on any host of a chain to manipulate the session, but there are only three positions with different effects. We assume H_0 is the intruder's host, H_1 is our MP, and H_{n+1} is the victim's host. Fig.3 shows the three positions: one is on any host (except the intruder's host) before the MP; the second one is on the MP; and the third one is on any host (except the victim's host) after MP. In Fig.3, the solid black arrow represents the Send direction in a connection chain, and the dash one represents the Echo direction in the same chain.

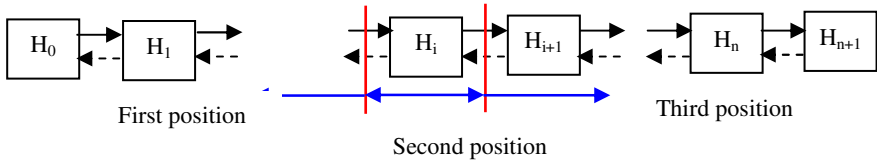


Fig. 3. A connection chain sample

If time perturbation is implemented on any host before the MP, it does not affect the Clustering-Algorithm to detect stepping-stone intrusion at all. We use symbol $S^{(i,1)}$, $S^{(i,2)}$ to represent the Send sequence in the incoming connection and outgoing connection of the host H_i respectively. Similarly, $E^{(i,1)}$ and $E^{(i,2)}$ are used to represent the corresponding Echo sequence. Before any time perturbation, the send and echo packets captured on H_i , and H_i in a certain period of time are as sequence set (1) shows.

$$\begin{aligned}
 S^{(1,1)} &= \{s_1^{(1,1)}, s_2^{(1,1)}, \dots, s_n^{(1,1)}\}, & S^{(i,1)} &= \{s_1^{(i,1)}, s_2^{(i,1)}, \dots, s_n^{(i,1)}\} \\
 S^{(1,2)} &= \{s_1^{(1,2)}, s_2^{(1,2)}, \dots, s_n^{(1,2)}\}, & S^{(i,2)} &= \{s_1^{(i,2)}, s_2^{(i,2)}, \dots, s_n^{(i,2)}\} \\
 E^{(1,1)} &= \{e_1^{(1,1)}, e_2^{(1,1)}, \dots, e_m^{(1,1)}\}, & E^{(i,1)} &= \{e_1^{(i,1)}, e_2^{(i,1)}, \dots, e_m^{(i,1)}\} \\
 E^{(1,2)} &= \{e_1^{(1,2)}, e_2^{(1,2)}, \dots, e_m^{(1,2)}\}, & E^{(i,2)} &= \{e_1^{(i,2)}, e_2^{(i,2)}, \dots, e_m^{(i,2)}\}
 \end{aligned} \tag{1}$$

Here, each element represents the timestamp of a Send or an Echo. For simplifying our analysis, we assume that there is no packet drop, combination, insertion, and generation. Actually removing all these assumptions does not affect our analysis result. If there is time perturbation in the outgoing connection of H_1 , most probably this perturbation can propagate to host H_i . We use Δ_i to represent the perturbation over packet s_i . The perturbed sequences in H_1 , and H_i are as sequence set (2) shows. From sequence set (1), we know for any Send, such as $s_k^{(i,2)}$ in the outgoing connection of H_i , assuming its Echo is $e_k^{(i,2)}$, its RTT is $e_k^{(i,2)} - s_k^{(i,2)}$. From sequence set (2), even though the k^{th} Send has been perturbed to $s_k^{(i,2)} + \Delta_k$, but the corresponding Echo is affected to $e_k^{(i,2)} + \Delta_k$ as well, so the RTT keeps the same. Our conclusion is that time perturbation on any host before the MP does not affect the performance of the

Clustering-Algorithm. It holds the same conclusion to the case when a time perturbation is implemented in the host MP.

$$\begin{aligned}
S^{(1,1)} &= \{s_1^{(1,1)}, s_2^{(1,1)}, \dots, s_n^{(1,1)}\} \\
S^{(1,2)} &= \{s_1^{(1,2)} + \Delta_1, s_2^{(1,2)} + \Delta_2, \dots, s_n^{(1,2)} + \Delta_n\} \\
E^{(1,1)} &= \{e_1^{(1,1)} + \Delta_1, e_2^{(1,1)} + \Delta_2, \dots, e_m^{(1,1)} + \Delta_n\} \\
E^{(1,2)} &= \{e_1^{(1,2)} + \Delta_1, e_2^{(1,2)} + \Delta_2, \dots, e_m^{(1,2)} + \Delta_n\} \\
S^{(i,1)} &= \{s_1^{(i,1)} + \Delta_1, s_2^{(i,1)} + \Delta_2, \dots, s_n^{(i,1)} + \Delta_n\} \\
S^{(i,2)} &= \{s_1^{(i,2)} + \Delta_1, s_2^{(i,2)} + \Delta_2, \dots, s_n^{(i,2)} + \Delta_n\} \\
E^{(i,1)} &= \{e_1^{(i,1)} + \Delta_1, e_2^{(i,1)} + \Delta_2, \dots, e_m^{(i,1)} + \Delta_n\} \\
E^{(i,2)} &= \{e_1^{(i,2)} + \Delta_1, e_2^{(i,2)} + \Delta_2, \dots, e_m^{(i,2)} + \Delta_n\}
\end{aligned} \tag{2}$$

However, if time perturbation is performed on any host after the MP, the RTTs are affected, but the downstream length estimated by the Clustering-Algorithm remains the same. We assume that the host H_{i+k} is manipulated by an intruder, and the time perturbation for each Send is bounded by Δ . Obviously, for any Sends, their RTTs are not affected by the time perturbation conducted on H_{i+k} until the connection chain goes through H_{i+k} . The RTTs are affected when the chain connects to any host after H_{i+k} . But we still could distinguish two levels of the connection chain by observing the RTTs under the condition that the time perturbation is bounded. We assume there are two hosts H_u, H_v after H_{i+k} , and $i+k < u < v < n+l$. When the chain is extended to H_u , the Send and Echo sequence captured on the MP is as sequence set (3) shows. Sequence set (4) shows the Send and Echo sequence when the chain is extended to H_v .

$$\begin{aligned}
S^{(i,2,u)} &= \{s_1^{(i,2,u)}, s_2^{(i,2,u)}, \dots, s_n^{(i,2,u)}\} \\
E^{(i,2,u)} &= \{e_1^{(i,2,u)} + \Delta_{u1}, e_2^{(i,2,u)} + \Delta_{u2}, \dots, e_m^{(i,2,u)} + \Delta_{um}\}
\end{aligned} \tag{3}$$

$$\begin{aligned}
S^{(i,2,v)} &= \{s_1^{(i,2,v)}, s_2^{(i,2,v)}, \dots, s_n^{(i,2,v)}\} \\
E^{(i,2,v)} &= \{e_1^{(i,2,v)} + \Delta_{v1}, e_2^{(i,2,v)} + \Delta_{v2}, \dots, e_m^{(i,2,v)} + \Delta_{vm}\}
\end{aligned} \tag{4}$$

Here we use u , and v as the superscript to distinguish two different hosts, and as the subscript to distinguish different time perturbation. Even though the time perturbations on host H_{i+k} are different for different Sends, but all the perturbations are assumed to be bounded by Δ . For a Send q , its RTT under the cases (3) and (4) are computed as the following, respectively.

$$\begin{aligned}
RTT_{(q,u)} &= e_q^{(i,2,u)} + \Delta_{uq} - s_q^{(i,2,u)} = RTT'_{(q,u)} + \Delta_{uq} \leq RTT'_{(q,u)} + \Delta \quad (a) \\
RTT_{(q,v)} &= e_q^{(i,2,v)} + \Delta_{vq} - s_q^{(i,2,v)} = RTT'_{(q,v)} + \Delta_{vq} \leq RTT'_{(q,v)} + \Delta \quad (b)
\end{aligned} \tag{5}$$

Here $RTT'_{(q,u)}, RTT'_{(q,v)}$ represent the RTT before any perturbation conducted on host H_{i+k} . From (5a) and (5b), we see it is still able to distinguish $RTT_{(q,u)}$ from $RTT_{(q,v)}$.

Even though the RTTs are changed after time perturbation, but we are still able to estimate the number of connections of a chain by applying the Clustering-Algorithm.

5 Resistance to Chaff Perturbation

Another way to evade detection is chaff perturbation, which means an intruder could manipulate an outgoing connection by inserting some meaningless packets into the original Send sequence. This section is to analyze the resistance of the Clustering-Algorithm to intruders' chaff perturbation. Most probably an intruder would manipulate a session by combining time and chaff perturbation together, rather than only any one of them. But here we only focus on analyzing the resistance analysis to chaff perturbation. To simplify our analysis, we make the following two assumptions:

- 1) An intruder could insert meaningless packets into any two consecutive Sends, but the timestamps of the original Sends cannot be changed;
- 2) Whatever how many meaningless packets are inserted, the meaningful packets are still the majority.

We define inserting ratio as the proportion between the number of inserted packets and the number of whole packets including chaff and original packets. Assumption 2) indicates the inserted packets cannot be the majority. If we have n normal Sends, the number of the inserted packets among the n Sends cannot be more than n . So assumption 2) indicates that the inserting ratio is bounded by 50% in chaff perturbation.

For an interactive session, the inputs are supposed to be executed at the victim site. Any chaff should be removed before it reaches the victim site, otherwise it would affect the execution of an input command. Depending on the different locations where an intruder sets up and removes a chaff perturbation, our analysis focuses on the following situations: 1) a chaff attack is set up before a MP (included), and removed before the MP (included); 2) a chaff attack is set up before a MP (included), and removed at any host after the MP; 3) a chaff attack is set up at any host after the MP, and removed at any host before the victim site.

We detect stepping-stone intrusion with the Clustering-Algorithm by recognizing the significant clusters which are the clusters with significant clustering rate. The study of the Clustering-Algorithm about the resistance to intruders' chaff perturbation is equivalent to the study of clustering rate affected by chaff perturbation.

The clustering rate is not affected if a chaff attack is set up before a MP, and removed at any host before the MP. We monitor and capture all the Sends and Echoes at the host MP, so the Send and Echo sequences are not affected any more by the chaff perturbation. Obviously this attack cannot affect the performance of the Clustering-Algorithm in detecting stepping-stone intrusion. Similar to this case, the clustering rate is also not affected if a chaff attack is set up at any host after the MP and removed the chaff at any host before the victim site. The Send sequence at the MP cannot be changed because the chaff attack is after the MP, as well as the Echo sequence because the chaff packets have been removed before they are echoed and the timestamps of the original Sends cannot be changed based on our first assumption.

However, the clustering rate would be affected if a chaff perturbation was set up before a MP (included), and removed at any host after the MP. Our study shows that the Clustering-Algorithm can still detect stepping-stone intrusion if the inserting ratio

of a chaff perturbation is bounded by 50%. Similar to what Fig.3 shows, we still assume that H_i is the MP, H_0 is the intruder's host, and H_{n+1} is the victim's host. Because the chaff is set up at any host before H_i , comparing to the original Send sequence S of the outgoing connection in H_i , the current Send sequence is S' and the Echo sequence is E' .

$$S = \{s_1, s_2, \dots, s_n\}$$

$$S' = \{s_1, c_{(1,1)}, \dots, c_{(1,k)}, s_2, \dots, s_{n-1}, c_{(n-1,1)}, \dots, c_{(n-1,q)}, s_n\}$$

$$E' = \{e_1, e_2, \dots, e_m\}$$

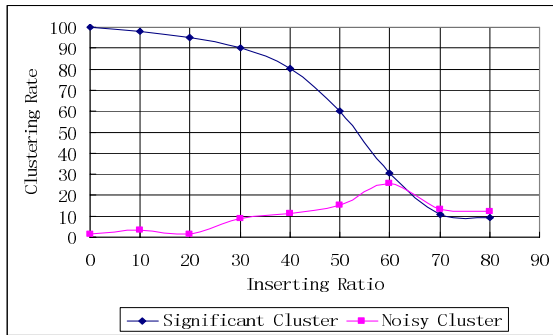


Fig. 4. The Affection to clustering rate

Here we use $c_{(i,j)}$ to represent the timestamp of a chaff packet. We use S and E' to find the significant clusters if without chaff perturbation. But with the chaff perturbation, we need to use S' and E' to find the significant clusters. This will affect the performance of the Clustering-Algorithm in two aspects. First, if the timestamps of the chaff are very close to either the Send before them or the Send after them, this only affects the clustering rate of each significant cluster without affecting the number of significant clusters. Second, if the timestamps of the chaff were not close to the original Sends, it would form one or more new clusters with probably a high clustering rate, and further affect the number of the significant clusters. In this case, the performance of the Clustering-Algorithm would be affected. But the experimental simulation result showed that the Clustering-Algorithm could still estimate the downstream length accurately under the situation that the chaff ratio is bounded by 50%. Fig.4 and Fig.5 show the affection to the clustering rate and to the estimation of the downstream length, respectively.

We did the simulation as the following. First, we monitor a connection chain from its start to its end at our MP. Its downstream ending length is 5 connections. We capture the whole Sends and put them into sequence S , as well as Echoes into E . Then we form S' by randomly inserting some packets into S , but control the inserting ratio to be 10%, 20%, ..., 80% respectively. We use the Clustering-Algorithm with input S' and E to estimate the downstream length and study the affection to the

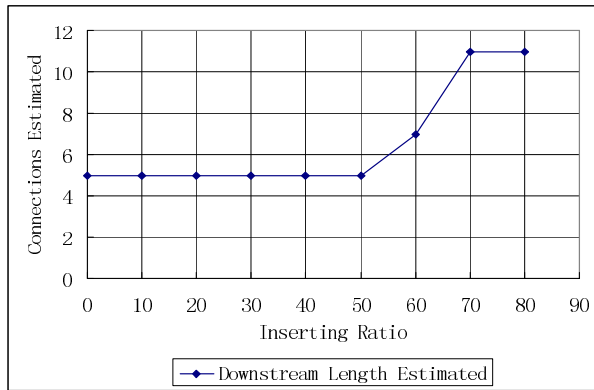


Fig. 5. The Affection to downstream length estimation

clustering rate of a specific significant cluster, and the affection to the downstream length estimation with the variation of the inserting ratio. The Simulation result showed that we can still distinguish a significant cluster from the other noise clusters and the Clustering-Algorithm can work well in detecting stepping-stone intrusion if the inserting ratio is bounded by 50%. In Fig.4, the clustering rate varies upon the variation of inserting ratio. The blue curve in Fig.4 represents variation between the clustering rate of a significant cluster and the inserting ratio. And the purple curve represents the variation between the biggest clustering rate of other non-significant clusters and the inserting ratio. The bigger the gap between the above two clustering rate, the easier to distinguish them. Fig.4 shows that this gap is as high as 45% when the inserting ratio is 50%, but it decreases to as low as no more than 5% when the inserting ratio is increased to 60%. Fig.5 shows that the downstream length is 5 connections that corresponds the real downstream length of the chain under the situation that the chaff ratio is no more than 50%. However, if the chaff ratio is more than 50%, the downstream length estimated is not accurate. As Fig.5 shows, if the chaff ratio is as high as 70%, the downstream length estimated by the Clustering-Algorithm would be 11 connections which is 6 connections more than the real case. From this simulation results, we conclude that the Clustering-Algorithm can resist intruders' chaff perturbation to a certain extent.

6 Related Work

The related work can be classified into two categories: a) detecting stepping-stone; b) detecting stepping-stone intrusion. Category a) includes the approaches proposed in papers [2], [1], [3], [5], [6]; Category b) includes the algorithms proposed in papers [4], [7], [8], [9]. The difference between the two categories is that the approaches in category a) can only detect if a host is used as a stepping-stone, but the approaches in category b) can detect not only a host is used as a stepping-stone, but also if the host is used by an intruder. Being used as a stepping-stone does not mean being used by an intruder because some legitimate users (applications) also need a host to be used as a stepping-stone. So there is one problem to judge if host is used by an intruder. Most

probably the approaches used to detect a stepping-stone intrusion are to exploit a common sense that some legitimate applications (users) may use a host as a stepping-stone, but the number of the compromised hosts may not be too much. If a user could connect to an end host directly, there is no reason to access the end host indirectly via some stepping-stones. So the key in the approaches of category b) is to estimate the number of compromised hosts (networks) from a MP.

To determine if a host is used as a stepping-stone is easier than to determine if a host is used by an intruder. Most approaches in category a) are to compare an incoming connection with an outgoing connection, such as content-based method [2], time-based method [1], [3], packet-number-based method [6]. They have a problem of being vulnerable to intruders' evasion except the method in paper [6]. And they all have a common problem which is high false alarm rate in detecting stepping-stone intrusion. The approaches in category b) can overcome this disadvantage, but they all suffer from estimating a downstream length imprecisely and being vulnerable in detecting intruders' evasion.

7 Conclusions and Future Work

In this paper, a Clustering-Algorithm has been proposed to predict stepping-stone intrusion. Comparing with the previous approaches, it has the advantage of detecting stepping-stone intrusion without false alarms and low misdetection. Moreover, the analysis and simulation results showed that the Clustering-Algorithm can resist an intruder's time perturbation completely, as well as chaff perturbation to a certain degree.

The problems of this algorithm is 1) we need to monitor a connection chain from the start to the end; 2) to obtain zero misdetection, we need to assume that a MP must be close to an intruder's site. The reason for second problem is that the Clustering-Algorithm only estimates the downstream length of a connection chain. To overcome this, in the future, we need to design a new algorithm which can estimate both the downstream length and the upstream length of a connection chain to avoid misdetection.

References

- [1] Yin Zhang, Vern Paxson: Detecting Stepping-Stones. Proceedings of the 9th USENIX Security Symposium, Denver, CO, August (2000) 67-81.
- [2] S. Staniford-Chen, L. Todd Heberlein: Holding Intruders Accountable on the Internet. Proc. IEEE Symposium on Security and Privacy, Oakland, CA, August (1995) 39-49.
- [3] K. Yoda, H. Etoh: Finding Connection Chain for Tracing Intruders. Proc. 6th European Symposium on Research in Computer Security (LNCS 1985), Toulouse, France, September (2000) 31-42.
- [4] Kwong H. Yung: Detecting Long Connecting Chains of Interactive Terminal Sessions. Proceedings of International Symposium on Recent Advance in Intrusion Detection (RAID), Zurich, Switzerland, October (2002) 1-16.
- [5] D. L. Donoho (ed.): Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. Proceedings of International Symposium on Recent Advances in Intrusion Detection, Zurich, Switzerland, September (2002) 45-59.

- [6] A. Blum, D. Song, And S. Venkataraman: Detection of Interactive Stepping-Stones: Algorithms and Confidence Bounds. Proceedings of International Symposium on Recent Advance in Intrusion Detection (RAID), Sophia Antipolis, France, September (2004) 20-35.
- [7] J. Yang, S-H S. Huang: A Real-Time Algorithm to Detect Long Connection Chains of Interactive Terminal Sessions. Proceedings of 3rd International Conference on Information Security (Infosecu'04), Shanghai, China, November (2004) 198-203.
- [8] J. Yang, S-H S. Huang: Matching TCP Packets and Its Application to the Detection of Long Connection Chains. Proceedings of 19th IEEE International Conference on Advanced Information Networking and Applications (AINA'05), Taipei, Taiwan, March (2005) 1005-1010.
- [9] J. Yang, S.H. S. Huang: Characterizing and Estimating Network Fluctuation for Detecting Interactive Stepping-Stone Intrusion. The proceedings of 3rd International Conference on Communication, Network and Information Security, Phoenix, Arizona, November (2005) 70-75.
- [10] University of Southern California: Transmission Control Protocol. RFC 793, September (1981).
- [11] M. Friedman, A. Kandel: Introduction to Pattern Recognition: Statistical, Structural, Neural, and Fuzzy Logic Approaches, NJ World Scientific Publishing Co., River Edge, London, (1999).
- [12] B. Mirkin: Mathematical Classification and Clustering, Kluwer Academic Publishers, Dordrecht, The Netherlands, (1996) 169-198.
- [13] A. Jain, R. Dubes: Algorithms for Clustering Data, Prentice Hall, Inc., New Jersey, (1988) 55-143.

A Wireless Intrusion Detection System for Secure Clustering and Routing in Ad Hoc Networks

Luciano Bononi and Carlo Tacconi

Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, Italy
{bononi, ctacconi}@cs.unibo.it

Abstract. Intrusion detection and secure routing schemes have been proposed for increasing the security and reliability in critical scenarios like mobile ad hoc networks. In this paper we present an integrated secure routing system based on Intrusion Detection Systems (IDS) and SUCV (Statistically Unique and Cryptographically Verifiable) identifiers. The proposed IDS has been used for the support of secure AODV routing, named IDS-based Secure AODV (IS-AODV), in a wireless ad hoc network scenario. Our IDS solution is based on the detection of behavior anomalies on behalf of neighbor hosts, with passive reactions, aiming to create a *cluster* whose route paths will include only safe nodes, eventually. Simulation results show that the proposed IDS is effective in isolating misbehaving hosts, and it assists the AODV secure routing scheme to converge in finding end-to-end safe routes.

Keywords: Intrusion Detection System, Secure Clustering and Routing, Statistically Unique and Cryptographically Verifiable Identifiers, Mobile Ad Hoc Networks.

1 Introduction

In Mobile Ad Hoc Networks (MANETs) the set of dynamically interacting mobile hosts should cooperate and each one should act like a router to enable end-to-end communications along a multi-hop dynamic routing path of radio links. The MANET scenario is considered a stressing scenario for routing protocols due to hosts' mobility, causing frequent route updates and link failures. Several multi-hop routing protocols for ad hoc networks have been proposed: most popular ones include DSR [4], OLSR [2], DSDV [3], AODV [5]. A majority of these protocols relies on the assumption of a trustworthy cooperation among all participating devices: unfortunately, this may not be a realistic assumption in real systems. Malicious nodes could exploit characteristics of MANETs to launch various kinds of attacks. Conventional methods and infrastructures for hosts' identification and authentication may not be available on MANETs, since the availability of a Certification Authority (CA) or a Key Distribution Center (KDC) cannot be always assumed over dynamic and infrastructureless networks. Many *Intrusion Detection System*

(IDS) solutions have been proposed for wired networks, based on monitoring of real-time traffic at statically defined strategic points: switches, gateways, and routers. In MANETs, nodes' mobility cannot be restricted in order to let the IDS to operate or collect data. Security solutions have been proposed at the routing layer in MANETs. As an example, *Secure Routing* schemes have been designed to incorporate security features into routing protocols. Several architectures and detection mechanisms for realizing integrated IDSs and secure routing protocols have been proposed for MANETs. They will be sketched in the Appendix A.

In this paper, we illustrate the design of a secure routing protocol (based on AODV), called IDS-based Secure AODV (IS-AODV), implemented by adopting an IDS solution and the concept of *Statistically Unique and Cryptographically Verifiable* (SUCV) identifiers [20] [21], [25], for IPv6-based MANETs. IS-AODV is based on SUCV and mutual verification of nodes' behaviors during the path creation processes. SUCV identifiers are used to realize a secure binding between IPv6 addresses and cryptography keys, without requiring any trusted CA or KDC. To the best of our knowledge, different solutions incorporating the same concepts have been proposed in SecAODV [22], and DSR-based solutions [25] concerning the SUCV adoption, and in the *watchdog* solution [9], concerning the *mutual verification* concept. IS-AODV is basically different from secAODV [22], because it uses the IDS as the basis for implementation of secure AODV routing. In order to control the behavior of neighbors during the route discovery and data forwarding phases, in IS-AODV, each node monitors the traffic whose path includes the node itself: when a node N perceives a suspect behavior from a neighbor host, the IDS reaction is *passive*: the information is not advertised to local nodes, and the node N does not rely/assist any suspect neighbor node communication. In this way, only safe routes will survive in the route creation and route maintenance processes, and a *cluster* would emerge that will eventually include only safe nodes. In addition, unlike secAODV, the proposed IS-AODV scheme does not require any cryptography operation in the *intermediate* nodes. IS-AODV is different from the *watchdog* solution proposed in [9], because our mechanism is designed to defend a Distance Vector routing protocol (like AODV), while the watchdog is created to defend a Source Routing protocol (like DSR). Basically, the watchdog node must know where a packet will be in two hops: this information is given with the DSR protocol, but not with distance vector protocols, like AODV. For this reason, the IS-AODV mechanism introduces one low-overhead additional field for standard AODV Route Request (RREQ) and Route Reply (RREP) messages (see section 2.3). In addition, a public key cryptography (or more lightweight symmetric cryptography, after a safe path is found), is used to verify the signature added in routing and data packets, only by end-nodes. This allows the end-to-end packet verification. Moreover, unlike the watchdog mechanism, IS-AODV does not accuse a node if it drops a packet: *i*) because this may happen due to collisions, and *ii*) because the node realizing a drop-attack would be self-excluded from the path creation process. To summarize, IS-AODV is different with respect to the watchdog mechanism, and other cooperation enforcement schemes like CORE [18] and CONFIDANT

[19], because *i*) IS-AODV is realized to defend a Distance Vector protocol like AODV, *ii*) the information about corrupted or safe nodes is not advertised to other nodes and *iii*) the spoofing attack has not critical effects [24]. The effect of MAC layer collisions on IS-AODV is discussed in section 3.

The paper structure is the following: in Section 2 we illustrate the design, implementation and assumptions of the proposed IDS mechanism, in Section 3 we illustrate a simulation model and results obtained, and in Section 4 we draw some conclusions. In the Appendix we sketch the state of the art in secure routing protocol solutions and IDSs. A discussion and solutions to possible attacks considered for this system can be found in [24].

2 The Security System Design

At the network layer, we can assume that a MANET is defined as the set of cooperating nodes adopting a common routing protocol. Under this assumption, it is possible for safe nodes to assume the routing protocol specification as a common set of guidelines representing the *normal behavior*. Every node diverging from the normal behavior is locally considered *unsafe*. So, the proposed *anomaly-based detection mechanism* allows the detection of many types of attacks, by defining an attack (or anomalous behavior) as “*a different behavior than the one defined by homogeneous protocol specifications*”.

2.1 Design Goals and System Assumptions

The proposed security system is realized to achieve the following objectives. As described in [1], an IDS for ad hoc networks:

g1) should not introduce new weaknesses in the system, that is it should ensure self-integrity without enabling new attack directions;

g2) should need few system resources and should not degrade the system performances by introducing significant computation and communication overheads;

g3) should be always on in background, transparently to the users.

In addition, under the routing viewpoint, we define the additional objectives:

g4) end-to-end communications are performed only on safe routes: a safe route is a path realized by safe nodes, connecting two safe end-points;

g5) if a safe route exists between two safe end-points, it will be adopted, eventually;

g6) a cluster composed by safe nodes transparently emerges as a side effect of the IDS passive reactions: that is, without propagating any information about the node corruption;

g7) no need for cryptography operations in the intermediates nodes: only end-points implement cryptography functions (like in transport/application level services). This choice saves energy resources and increases communication efficiency in the system (see *g2*). This is even more important for battery-based and low computation-power portable devices;

g8) no need to identify the attack type, if any: the system activity simply preserves correct routing protocol specifications;

g9) support for end-to-end authentication, confidentiality and integrity.

The following assumptions are the basis for the proposed mechanism realized in a MANET environment:

- a1)* At the routing viewpoint, the end-points of a communication (source S and destination D) are implicitly safe;
- a2)* every link between the participating nodes is bidirectional;
- a3)* nodes operate in promiscuous mode at the MAC layer, meaning that nodes can listen to their neighbors' transmissions;
- a4)* all safe nodes have the IDS activated, unless they may be considered as malicious nodes;
- a5)* all system nodes (both safe and malicious ones) know a pre-defined one-way hash function, which characterize the MANET;
- a6)* the MANET is implicitly homogeneous under the AODV routing protocol viewpoint.

2.2 System Components and Definitions

The security system mechanism presented in this work to support a secure routing solution for the AODV protocol, is based on two main components:

- 1) the Intrusion Detection System (IDS), which is based on *host and anomalies detection* with *passive* reaction.
- 2) the Statistically Unique and Cryptographically Verifiable (SUCV) identifiers, to ensure a secure binding between IPv6 address and public key, without requiring any CA or KDC. If a safe route between two safe end-points exists, the aim of our mechanism is to find that route, eventually. On the other hand, the secondary aim of our scheme is to create an emerging cluster of safe nodes. To realize such aims, the first point is the safe route construction and maintenance process. This is obtained by exploiting:
 - i)* the AODV definition (of RREQ and RREP phases, see section 2.3),
 - ii)* the end-to-end authentication of source and destination nodes. This authentication is obtained by adopting a Public Key Cryptography scheme, where keys are bound to node identities by means of a SUCV mechanism,
 - iii)* the end-to-end signature verification for routing and data packets, to detect any malicious activity by corrupted nodes appearing in the originated path,
 - iv)* the IDS-based mutual observation and control of routing and data transmissions between neighbor hosts, to detect behavior anomalies.

To summarize, during a path-discovery process, each node belonging to the forming-path monitors the routing or data packets forwarded by other nodes, within one-hop distance, to detect anomalous behaviors. When the number of behavior anomalies exceeds a predefined threshold (see section 2.4), the misbehaving node is considered corrupted by the observer. In this case, the IDS reaction is *passive*, that is, the information about host corruption is not advertised to neighbor nodes. As a reaction, the accused node is not trusted and not assisted (undefinitely or temporarily, see 2.4) by the accusing node. The choice to adopt **passive reactions** against malicious nodes is motivated because *active* reactions would require some kind of distributed majority (voting or ranking)

procedure. This procedure may be very expensive, due to high number of service messages. In addition, all voting messages would need authentication obtained by cryptography operations. In absence of authentication, corrupted nodes could bias the majority or, under some scenarios, they could coordinate themselves to locally attack and control the voting process, and to accuse safe nodes. For these reasons, we decided to adopt the *passive* reaction, based on the “trust nobody” assumption: every safe node aims to create a *secure cluster* by exploiting only local (implicitly trusted) information obtained by sniffing packets forwarded by one-hop neighbors. After the creation and maintenance of the secure path between end-nodes, the data confidentiality, integrity, and authentication can be implemented by a more lightweight symmetric cryptography scheme.

In general, IS-AODV is independent by the cryptography schemes adopted: as an example, one-way hash functions (like MD5 or SHA-1) could be used for SUCV IDs, public key cryptography scheme (like ECC or RSA) and the symmetric cryptography scheme (like AES or DES) can be freely adopted according to the system, network and application requirements.

2.3 System Overview

The AODV routing algorithm is based on the flooding of Route Requests (RREQs) packets for Route Discovery phases, and unicast Route Reply (RREPs) packets for the route reply phase. Details can be found in [5]. Figure 1 shows the modules’ architecture of a node implementing the IDS. The IDS module captures all the

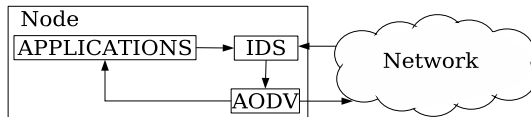


Fig. 1. Modules’ architecture of a node with the proposed IDS

node traffic from/to the MANET, and it filters packets that should be further processed by the AODV routing protocol. Every packet received by adjacent nodes during a path discovery process is checked, to verify if the packet is sent by a previously identified malicious node. Every packet received by an adjacent node in a pre-defined path, is checked to verify if it has been corrupted (like in watchdog mechanism). In both cases the packet is immediately discarded. In other words, malicious nodes are excluded from the paths they attempt to attack.

2.3.1 AODV Modifications

AODV has been slightly modified to implement IS-AODV:

- i*) RREQ and RREP message headers have been extended to include a pair of SUCV identifiers. The overhead introduced is quite marginal;
- ii*) the AODV routing table is made more connection-oriented, being enriched by including data structures about end-nodes, defined within the IDS mechanism. SUCV identifiers and the extension fields for AODV routing packets introduced by the IDS implementation will be defined in the following, together with the definition of the mutual verification scheme, and the IDS detection and reactions.

2.3.2 SUCV Identifiers

Differently from IP addresses, SUCV identifiers [20] are flat identifiers with no topology-related meanings. Basically, the SUCV IDs introduce the advantage of an implicit cryptography binding between a node’s identifier and its public key (or certificate). A node auto-configures its own Crypto-Based Identifier (CBID) by doing the following operations:

- 1) it creates a pair of public and private keys (P_K and PrK).
- 2) it creates its own CBID: $CBID = hash(P_K)$, where the hash function is a system configuration parameter (assumption *a5*).

In this way the key management is simplified, since no third parties need to be involved either in creating or in distributing the public keys. Provided that the bit-length of the CBID’s is large enough, these identifiers have two very important properties: they are *statistically unique* and *bound in secure way to a given node*. Any other node can easily verify the CBID signature without relying on any centralized security service, such as a Public Key Infrastructures (PKIs). The *authentication, confidentiality* and *integrity* (see *g9*) implemented in our target system is based on public key cryptography, and more specifically on the SUCV (or CBID) identifiers. When a node receives the $[CBID, P_K]$ pair (in a message header), it calculates the hash function of the public key P_K , and it compares the result with the received $CBID$: if the value is the same, the P_K will be used to decrypt the $\{DIGEST\}_{PrK}$ included into the RREQ or RREP.

2.3.3 RREQ Header Extension

To adopt the CBID identifiers we have extended the RREQ and RREP messages by adding some fields (see figure 2). With the IPv6 protocol, the CBID is half of the IPv6 address.

Original IPv6 RREQ		Original IPv6 RREP		
P_{KS}	$IPv6_{prev}$	P_{KD}	$IPv6_{prev}$	RREQ_ID
$\{DIGEST_{RREQ}\}_{PrK_S}$		$\{DIGEST_{RREP}\}_{PrK_D}$		SEQ_No

Fig. 2. RREQ and RREP extensions

In the RREQ message the following fields have been added:

P_{KS} : public key of the source node;

$\{DIGEST_{RREQ}\}_{PrK_S}$: Digest of the RREQ message, including the extension fields, excepted the destination sequence number and hop count value.

$IPv6_{prev}$: IPv6 address of the previous node propagating the RREQ during the path creation process between source S and destination D , used to implement the mutual verification of the path creation (see below).

The route discovery phase is based on public key fields to be included into the RREQ (and RREP) header extensions. In this way the IDS supports the

safe route creation over the most general scenario, without any assumption about any information sharing among MANET's nodes. Conversely, if a system information-sharing exists about the node identifiers and the related public key (as an example, by using an Hello-type broadcast message), the P_{KS} and P_{KD} could be removed in the RREQ and RREP extension fields, to reduce overheads. After the creation of the safe route-path, successive data transmissions are based on a more efficient and lightweight symmetric key mechanism. The symmetric key exchange could be realized just after the creation of the safe route, that is, without including symmetric keys in the headers of broadcasted RREQ and RREP messages.

2.3.4 Mutual Verification of the Path Creation

By looking at figure 3, the mutual verification process is explained: $IPv6_{prev}$ is set on a node to identify the previous node in the path (as indicated by arrows):

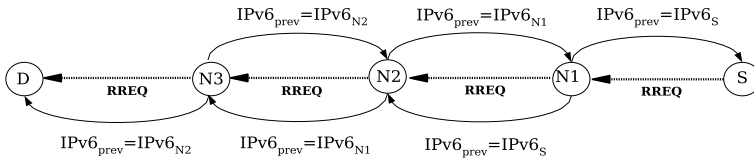


Fig. 3. $IPv6_{prev}$ setting

- 1) the intermediate node $N1$ will receive RREQ from source node S ;
- 2) upon reception of one RREQ from previous node S , $N1$ sets the $IPv6_{prev}$ value with $IPv6_S$. $N1$ then re-broadcasts the RREQ (which will propagate to neighbors $N2$ and S);
- 3) mutual verification of node S : S will detect the RREQ forwarded by $N1$. S checks that $IPv6_{prev} = IPv6_S$, and it compares last RREQ with its own cached RREQ copy. S determines in this way if $N1$ has correctly forwarded the RREQ, or not;
- 4) all intermediate nodes, including $N1$ and $N2$, do behave like S when propagating RREQs (and RREPs along the reverse path).

2.3.5 RREP Header Extension

In the RREP message the following fields have been added to the RREP header (see figure 2):

P_{KD} : public key of the destination node;

$IPv6_{prev}$: $IPv6$ address of the previous node along the return path;

$\{DIGEST_{RREP}\}_{Pr_{KD}}$: Digest of the RREP message, extension fields included;

$RREQ_ID$: copy of the RREQ ID value that has generated this RREP;

SEQ_No : copy of the RREQ's originator sequence number value that has generated this RREP.

The $IPv6_{prev}$ -based mutual verification management, as shown in figure 3, is repeated in the opposite forwarding direction for RREPs.

2.3.6 Conservative Property to Avoid the RREQ_ID and Sequence Number Attack Effects

The RREQ_ID and SEQ_No fields are used in RREPs to verify the definition of a safe route. This is required because the *RREQ ID attack* and the *sequence number attack* could be performed by byzantine nodes to interfere with the route construction. When an intermediate node N_I receives a RREP from N_{I+1} , it checks (in its own local log) if it previously overheard the corresponding RREQ forwarded in the opposite direction (from N_{I+1} to N_{I+2}). The checks performed include the correct IP settings, and the originator sequence number and RREQ ID corresponding to the RREP received. In addition, N_I checks if it has previously received and forwarded a RREQ from N_{I-1} (that is, its previous node in the path between N_I and the source node). If the two conditions are true, then N_I forwards the RREP back to N_{I-1} , by contributing in this way to define the bidirectional verified route path between S and D .

2.4 IDS Detection and Reactions

In this section, the IDS detection and reaction processes are defined. In previous section, we illustrated how each network node is assumed to monitor only the traffic whose transmission path (or route discovery phase) includes it as an intermediate node. The IDS management of RREQs and RREPs requires additional data structures. The IDS reaction is based on a counter $MB_{i,j}$ of malicious behaviors detected by node i for each neighbor j . This counter is similar to the “reputation” concept used in other works. When the counter $MB_{i,j}$ exceeds a predefined **threshold value, TV** ($TV = 3$ in our experiments) then the node i will indefinitely (or temporarily) consider node j as a corrupted node. The value of TV is related to the degree of security: e.g. $TV = 10$ would result in tolerant networks (low security level) while $TV = 1$ would result in low tolerant network (high security level). If network nodes are assumed to be possibly infected by malicious code (as an example, virus-like attacks), then they could be temporarily suspected. In this way, once a “suspect validity time” expires (this validity time value must be properly defined according to the network features) the node could be reconsidered during path formation processes.

The IS-AODV RREQ-management requires the maintenance of three sets of IDs, for each node N in a path:

First Nodes: the set of nodes from which N receives the first valid RREQ that must be forwarded, according to the AODV specifications;

Alternative Nodes: the set of nodes from which N receives an RREQ whose RREQ IDs was already processed (that is, RREQ copies);

Next Nodes: the set of nodes from which N should receive the RREP. These nodes are identified as the nodes that correctly forwarded an RREQ received from N (i.e. whose $IPv6_{prev}$ indicated N).

The IS-AODV RREP-management requires an additional set of IDs for each node, to be updated during every route discovery process:

Selected Paths: the set of nodes to whom N sent an RREP that must be forwarded back to S , according to the AODV specifications;

The purpose of the above mentioned sets will be illustrated in the following examples. Let's consider the following path where the arrows illustrate the direction of the target packet flow:

$$N_1 \rightarrow N_2 \rightarrow N_3$$

if N_1 sends a (routing or data) packet along the route to N_3 , then N_1 is assumed to overhear the packet forwarded by N_2 . If N_2 corrupts the packet, then N_1 can detect the corruption, by counting an anomalous behavior for node N_2 . When the counter of malicious behaviors for node N_2 exceeds a **threshold value** on N_1 , then N_1 will not interact with N_2 , by locally assuming a *link-break*, under the routing protocol viewpoint, that is, by excluding the malicious node N_2 from the route path. Corrupted nodes are considered the same way as nodes that moved away, unless they behave in a correct way. Anyway, this reaction is not enough, to create a safe path between two safe end-points. The following

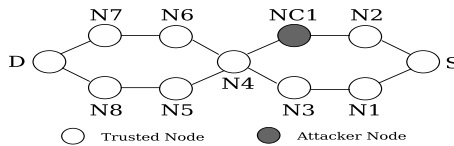


Fig. 4. Network with an attacker node

example illustrates how unsafe paths can be avoided by the IDS reactions. By looking at figure 4, let us assume that $NC1$ realizes a RREQ corruption, (for this case the *RREQ ID attack* is excluded), and that the RREP sequence from D to S , obtained after the route discovery process, is going to be forwarded on the following reverse path:

$$D \rightarrow N8 \rightarrow N5 \rightarrow N4 \rightarrow NC1 \rightarrow N2 .$$

Node $N2$ stops the RREP forwarding because it has checked the $NC1$ RREQ's corruption. For this reason, after a timeout, S will start a new route discovery process, by sending a new RREQ broadcast message. Note that malicious node $NC1$ still participates to this process. On the other hand, $N4$ (like each other node) maintains a *Selected Paths* set indicating the nodes where $N4$ has already forwarded a RREP message (that is, including $NC1$), and an *Alternative Nodes* set indicating the nodes where $N4$ has not still forwarded a RREP message. For this reason, upon reception of a new RREQ message from S to D , $N4$ will exclude nodes belonging to the *Selected Paths* set (that is, $NC1$). As an example, in figure 4, $N4$ will select $N3$ to receive the RREQ, and to respond later with the RREP message. When the *Alternative Nodes* set is empty, the *Selected Paths* set is restarted (that is, all nodes propagating RREQs are considered valid for forwarding RREPs). This management is implicitly derived by the concept of **AODV black list**, that we sketch in the following. When a node $N4$ forwards a RREP on a reverse path, it would expect to receive data packets, and not

another RREQ message from the path source S . If a node $N4$ receives another RREQ from the same source S to destination D , it may realize that:

- i*) at least one unidirectional (S to D) link is present between $N4$ and S , or
- ii*) at least one attacker node is present between $N4$ and S .

In both scenarios the RREP path must be (at least temporarily) discarded by node $N4$. If alternative paths exist, $N4$ will select one of these, as an example:

$$D \rightarrow N7 \rightarrow N6 \rightarrow N4 \rightarrow N3 \rightarrow N1 \rightarrow S.$$

The route validity and the identity of previous- and next-hop nodes in a valid path can be assumed by a node $N4$ when it receives an ACK (e.g. from $N6$) under a bidirectional TCP-like connection, or a Data packet (e.g. from $N3$) under a unidirectional UDP-like connection. While this happens, $N4$ would discard every other RREQ aiming to find a route between S and D . If the routing path explicitly expires on $N4$ due to AODV route entry expiration or a link-layer failure, then RREQs would be accepted again by $N4$ to recover the S-to-D path failure. Solutions for attacks to these policies can be found in [24].

2.4.1 Corrupted RREPs

If the destination node D receives a corrupted RREQ and no other safe RREQ has been received in the discovery process, then D will send an explicitly corrupted RREP anyway. This will allow to identify the current path as a path including at least a corrupted node, by causing the updates to the *Selected Path* set of intermediate nodes. In general, any corrupted RREP will not modify any routing table, and will be simply discarded by the source node S .

2.4.2 Conservative Property to Avoid Routing Loops

The following rule is implemented to avoid route loops: if a RREP is received from a node N that belongs to RREQ's *Alternative Nodes* set or *First Nodes* set, then N will be accused, because a node can receive RREPs only from nodes belonging to *Next Nodes* set of the corresponding RREQ message. For the same reason a data packet is forwarded from S to D only if it is received by the last node inserted into the *Selected Path* set.

2.4.3 AODV's Route Maintenance

The definition of AODV's issues on route maintenance, sequence numbers, Route Cache and Route Error management have been considered for IS-AODV and can be found in [24] due to space limitations.

2.5 Attacker Model

The attacker model considered is based on *active* and *internal* attack types. The following list illustrates a set of attacks described in [7],[8], that have been considered in the IDS and IS-AODV design. A short description of the IDS solution to contrast a given attack type is provided in [24]. The list of attacks includes: Black Hole, Route Disruption, Route Invasion, Modify and Forward, Denial of Service, End-node impersonation, Coordinated DOS attack, RREQ ID attack, IP attacks, Coordinated adjacent nodes' attack.

3 Performance Analysis

3.1 Introduction

To perform the performance investigation, a MANET system model has been implemented with the *ns2 Network Simulator*. The MAC layer model implements the IEEE 802.11 DCF protocol. The IS-AODV model is derived from the Uppsala University AODV model (AODV-UU)[23], freely available, and almost compliant with a real AODV implementation code. The complete IDS implementation has been modeled as an additional feature of MANET nodes.

3.2 Simulation Parameters and Metrics

Table 1 shows some most significant simulation parameters. It is worth noting that the speed of nodes is 10m/s on the average, and 20m/s maximum, with the pause time of 100 seconds under a Random Waypoint mobility model on a long rectangle-shaped area.

Table 1. Simulation parameters

Simulation duration	300 sec	Maximum Speed	20 m/s
Simulation Area	1500m * 300m	Packet rate	(CBR) 4 pkt/sec
Mobile hosts number	50	Host pause time	100 sec
Transmission Range	250 m	Connections Number	10, 20, 30
Movement model	Random waypoint	Corrupted nodes number	10 %, 25%, 50%

These assumptions may be considered quite unrealistic, but they have been defined in this way in order to stress the IDS and the routing mechanisms. All the attack types that have been implemented in the model, mainly against the RREQ and the RREP messages, are listed in [24]. Corrupted nodes try to establish connections with others nodes (both safe and corrupted ones). Our interest in the analysis is on connections originated between two safe end-points (as assumed in the mechanism's design). Our main interest is on effects and overheads of the proposed IDS and secure routing scheme (IS-AODV), with respect to standard AODV: *i*) the packet overheads introduced in RREQ and RREP packets, *ii*) the additional computation required for implementing the IDS and secure routing scheme, and *iii*) the number of additional RREQs needed to find a safe route, in a given scenario. The packet and computation overheads introduced can be considered marginal (see section 2). The proposed IDS system creates additional *virtual link breaks*, and route discovery processes, due to corrupted nodes' effects, and due to possible ambiguous effects of collisions and hidden terminals on the mutual verification mechanism. For these reasons, the main evaluation metric we will discuss here is the comparison of the average number of RREQs issued by a source node, needed to establish a end-to-end connection (under AODV), and a safe end-to-end connection (under IS-AODV). The *Average Number of RREQs* shown in the figures is the average number of RREQs that the source node must send to complete a route path creation (as a reaction to missing path or link

failures in existing paths). This evaluation will be obtained under different percentages of attacker nodes, and under different mobility and collision effects, in the MANET scenario:

- i) while the network is attacked and all the nodes were static during last 100 seconds, that is, when IS-AODV is active and nodes start to have good knowledge of their respective one-hop neighbors: *IS-AODV On (only safe routes), 100 sec. static scenario*;
- ii) while the network is attacked, IS-AODV is On, and mobile nodes cause more difficult secure-path creations: *IS-AODV On (only safe routes), steady-state mobile scenario*;
- iii) without attacker nodes, that is, by matching the standard AODV path formation process: *IS-AODV Off (pure AODV)*.

These three scenarios are compared to test the security system behavior under the attacks, with respect to ideal scenarios with no attacks (with standard AODV). We performed experiments with runs of 300 seconds of simulated time, and 50 nodes in the area. Results shown are within confidence intervals whose confidence level is 95%.

3.3 Simulation Results

In this section we present the obtained simulation results, with variable percentage of corrupted nodes, effects of mobility, collisions and hidden terminals for the performance index shown. Only most significant figures are shown due to space limitations. The figures 5.a (left) and 5.b (right) show the *Average Num-*

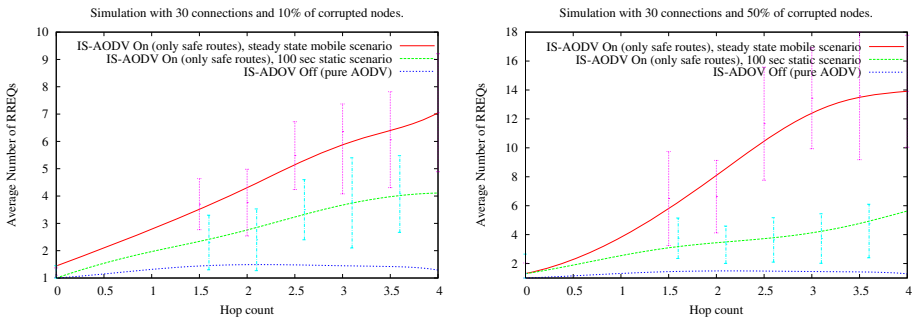


Fig. 5. Average Number of RREQs as a function of the mobility and hop distance

ber of RREQs needed for each active route-path under the AODV protocol in the MANET scenario with 50 active nodes, 30 active end-to-end connections (whose hop-count is indicated in the X axis). The percentage of corrupted nodes is 10% in figure 5.a and 50% in figure 5.b, respectively. By looking at the figures, the curves *IS-AODV Off (pure AODV)* show the average number of RREQs required by sources to find a generic route in the system, as a function of the path hop-count. The average value is comprised between 1 (that is, the optimal value) and 2. This slightly sub-optimal effect is produced by the mobility and

by some MAC-layer frame-loss effects (as an example, collisions due to RREQs broadcast storms on the destination node). The difference between static and mobile scenarios is marginal for this performance index (so only one is shown), because a path creation is fast enough to complete without significant effects of the modeled node mobility. The route-path length has marginal effect, and this indicates that one route path can be easily found within few attempts, given the simulated system characteristics. The curve *IS-AODV On (only safe routes), steady-state mobile scenario* show the same index above, as a function of the path length, in a steady-state scenario with mobile nodes, IS-AODV active, 10% attacker nodes in figure 5.a, and 50% attacker nodes in figure 5.b, respectively. The average number of RREQs to obtain a secure path increases with the path length, as expected, because the probability to have attackers in the candidate paths increases accordingly. The same consideration is valid in the comparison between 10% and 50% attacker nodes scenarios. The mobility effect of nodes contrasts the secure clustering that would emerge given the IDS effect in IS-AODV: nodes that locally accuse other nodes may move and lose this knowledge-base useful to create safe paths. The effect of the secure clustering is shown in the curve *IS-AODV On (only safe routes), 100 sec. static scenario*. The only difference with respect to previous curve is given by the static uniform distribution of nodes in the area. The average number of RREQs in the system reduces because nodes acquire more persistent information about neighbor nodes, and identify the attackers by excluding them in the current and future path creation processes. The average number of RREQs attempts that would be aborted during a path creation by the IS-AODV effects can be obtained as the difference between the IS-AODV On and IS-AODV Off curves.

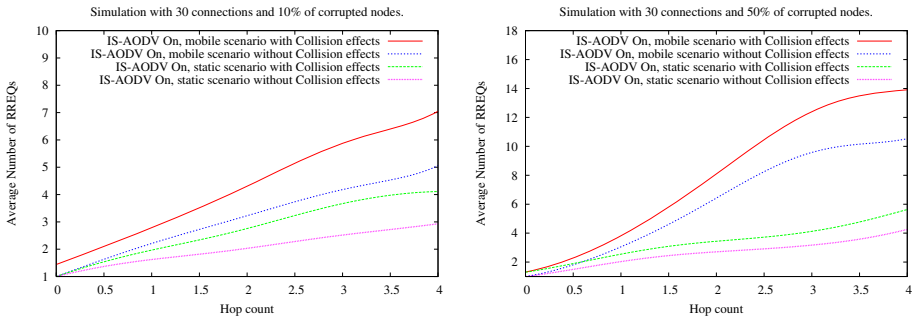


Fig. 6. Average Number of RREQs: hop distance and collision effects

The figures 6.a (left) and 6.b (right) show a comparison of the effects of collisions and hidden terminals on the IS-AODV mechanisms (10% and 50% corrupted nodes). The critical effects, under this viewpoint, are given by the signal collisions on the receivers, and by the hidden terminal effects on nodes implementing the mutual verification of path creation. A discussion of these critical effects can be found in the discussion of watchdog problems in [9]. As

an example, by looking at figure 4, if $N1$ is unable to sniff the RREQ sent by $N3$ to $N4$ (e.g. due to S transmitting) then $N1$ will not propagate the related RREP received from $N3$. This is due to the conservative design of our IDS. In IS-AODV, these effects translates in an increase in the number of RREQs needed to obtain a safe route. This overhead is shown in the curves *IS-AODV On, mobile scenario with collision effects* compared with *IS-AODV On, mobile scenario without collision effects* for the mobile scenario. In the static scenario, the overhead effect is shown in the curves *IS-AODV On, static scenario with collision effects* compared with *IS-AODV On, static scenario without collision effects*. We are currently working on less conservative IDS solutions to overcome this problem.

Additional results on the percentage of safe nodes that accuse a corrupted node, and the percentage of corrupted nodes that have been discovered by at least one safe node as a function of time is shown in [24].

4 Conclusions

In this work, we defined a new solution based on IDS and SUCV identifiers to assist the AODV routing protocol in finding end-to-end safe routes in a MANET scenario, called IDS-based Secure AODV (IS-AODV). The overhead represented by the number of RREQs during route discoveries and the impact of MAC layer collisions have been evaluated. Simulation results confirmed that the proposed IDS contributes to a transparent clustering of safe nodes, which isolate the attackers with a passive reaction.

References

1. P. Albers, O. Camp, J. M. Parcher, B. Jouga, L. Me, R. Puttini. Security in Ad Hoc Networks: a General Intrusion Detection Architecture Enhancing Trust Based Approaches. *WIS 2002. 4th Int'l Conf. on Enterprise Information Systems, 2002.*
2. T. Clausen, P. Jaquet, A. Laouti, P. Minet, P. Muhlethaler, A. Quyyum, L. Viennot, Optimized Link State Routing Protocol, *Internet Draft, draft-ietf-manet-olsr-06.txt, work in progress, Sep 2001.*
3. C. E. Perkins, P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, *Proc. of the SIGCOMM 94 Conference on Communications Architectures, Protocols and Applications, Aug. 1994.*
4. D. B. Johnson, D. A. Maltz, Y-C Hu, J. G. Jetcheva, The dynamic Source Routing Protocol for Mobile Ad hoc Networks (DSR), *Internet Draft, draft-ietf-manet-dsr-07.txt, work in progress, Feb 2002.*
5. C. Perkins, E Belding-Royer, Ad hoc On-demand Distance Vector (AODV), *Request For Comments (RFC) 3561, July 2003.*
6. S. Bhargava and D. P. Agrawal. Security Enhancements in AODV protocol for Wireless Ad Hoc Networks. *in Proc. of Vehicular Technology Conference, 2001.*
7. H. Deng, W. Li and D. P. Agrawal. Routing Security in Wireless Ad Hoc Networks. *IEEE Communications, October 2002.*

8. P. Ning and K. Sun. How To Misuse AODV: A Case Study of Insider Attacks against Mobile Ad-Hoc Routing Protocols. in *Proceedings of the 4th Annual IEEE Information Assurance Workshop*, pages 60-67, West Point, June 2003.
9. S. Marti, T.J. Giuli, Kevin Lai and Mary Baker. Mitigating Routing Misbehavior in Mobile Ad-hoc Networks. in *Proceedings of the 6th Annual ACM/IEEE international Conference on Mobile Computing and Networking*, pp. 255-265, 2000.
10. A. Patcha and A. Mishra. Collaborative Security Architecture for Black Hole Attack Prevention in Mobile Ad Hoc Networks. in *Proceedings of the Radio and Wireless Conference, RAWCON 2003*.
11. Yongguang Zhang, Wenke Lee and Yi-An Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *Wireless Networks*, Vol. 9 Issue 5, Sep. 2003.
12. C. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe and K. Levitt. A Specification-based Intrusion Detection System for AODV. *Proc. of the 1st ACM workshop on Security of ad hoc and sensor networks*, pp. 125-134. 2003.
13. J. Undercoffer and Anupam Joshi. Neighborhood Watch: An Intrusion Detection and Response Protocol for Mobile Ad-hoc Networks. *Tech Rep, UMBC, Oct. 2002*.
14. P. Papadimitratos, Z. J. Haas, Secure Routing for Mobile Ad hoc Networks , *Proceedings of the SCS Communication Networks and Distributed Systems, Modelling and Simulation Conference (CNDS 02)*, pp. 27-31, January 2002.
15. Y. C. Hu, A. Perrig, D. B. Johnson, Ariadne: A Secure On-demand Routing Protocol for Ad hoc Networks , *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom 02)*, pp. 12-23, September 2002.
16. B. Dahill, B. N. Levine, E. M. Royer, C. Shields, A Secure Routing Protocol for Ad hoc Networks , *Tech Rep, UM-CS-2001-037, Univ. of Massachusetts, Aug 2001*.
17. Y. C. Hu, D. B. Johnson, A. Perrig, SEAD: Secure Efficient Distance Vector Routing for Wireless Ad hoc Networks , *Proceedings of the 4th IEEE Workshop on mobile Computing Systems and Applications (WMCSA 02)*, pp. 3-13, June 2002.
18. P. Michiardi, R. Molva. Core: A Collaborative REputation mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. *Proceedings of IFIP Communication and Multimedia Security Conference 2002*.
19. S. Buchegger, J-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) *MobiHoc 2002*.
20. G. Montenegro and C. Castelluccia. Crypto-Based Identifiers (CBIDs): Concept and Applications. *ACM transaction on information and system security*, vol. 7 num. 1, February 2004, page 97-127
21. C. Castelluccia and G. Montenegro. Protecting AODV against Impersonation Attack. *ACM Mobile Computing and Communications Review*, vol 6 no 3, 07/2002
22. Anand Patwardhan, Jim Parker, Anupam Joshi, Michaela Iorga and Tom Karygiannis. Secure Routing and Intrusion Detection in Ad Hoc Networks. *Proc. of PerCom 2005*.
23. Uppsala University AODV implementation
<http://www.docs.uu.se/docs/research/projects/scanet/aodv/aodvuu.shtml>
24. Technical Report (IS-AODV)
http://www.cs.unibo.it/~bononi/Publications/is-aodv_tr_2006.pdf
25. R. B. Bobba, L. Eschenauer, V. D. Gligor, and W. Arbaugh. Bootstrapping Security Associations for Routing in Mobile Ad-Hoc Networks. In *proc. IEEE Global Telecommunications Conference*, December 2003.

Appendix

A Related Works

A.1 Secure Routing Protocol

Recent solutions for implementing secure routing protocols and IDSs over mobile ad hoc networks can be found in this section. The Secure Routing Protocol (SRP) proposed in [14] is based on DSR protocol [4], and contrasts the malicious behavior that may be originated by the discovery process of topological information. The basic assumption of SRP is that any two *end-to-end* nodes of a communication process would have a preliminary security association. Accordingly, SRP does not require: *i*) that any of the *intermediate* nodes perform cryptography operations, and *ii*) that intermediate nodes have a prior security association with the end nodes. ARIADNE [15] is a secure on-demand routing protocol that relies on highly efficient *symmetric* cryptography solutions. The ARAN [16] mechanism prevents modification, impersonation and fabrication attacks through the implementation of message authentication, integrity and non-repudiation mechanisms. SEAD [17] supports DSDV routing [3], and it is a robust solution against multiple uncoordinated attackers aiming to create incorrect routing information in other nodes. The protocol uses efficient *one-way hash functions* and it does not use asymmetric cryptography operations.

A.2 Intrusion Detection Systems

In [9], two techniques to improve throughput in ad hoc networks are described: *i*) in an effort to enforce the node cooperation, and *ii*) to detect the presence of byzantine nodes that fails to forward packets. The proposed solution is based on the implementation of *watchdogs* that identify misbehaving nodes, and a *pathrater* that helps end-nodes to avoid malicious nodes in the routing path choice. In [10] nodes are classified into *trusted* and *ordinary* nodes, and a watchdog mechanism is executed on the trusted nodes. Every node that subsequently joins the network has to prove its trustworthiness to be admitted to the trusted group. The main assumption in [10] is that any node will always behave in trusted or malicious way, indefinitely. In [6], an IDS prevents attacks by implementing an Intrusion Detection Module (IDM) and an Intrusion Response Module (IRM). The IRM is based on a local counter (for every node i) $C_{i,j}$ respectively associated to any other neighbor node j , that is incremented whenever a malicious act of node j is encountered. When the $C_{i,j}$ value reaches a predefined threshold then the suspect-warning about the node j is actively propagated to the entire network by node i . In [11], individual IDS agents are placed on every node, to monitor all local activities (including user and system side activities). When an IDS agent detects a local intrusion, it initiates a global response: all IDS agents will cooperatively participate in global intrusion detection actions to isolate the corrupted node. In [12] the system uses *Network Monitors* distributed on a subset of selected nodes into the network, to detect attacks against AODV

routing. In [13] the system uses an IDS based on neighbor node's snooping of packets transmissions: a node hearing two consecutive transmissions, along the path from source to destination, checks that the packet and its route information is not modified in flight by malicious nodes. This approach uses two modes of operation: *i) passive* (to protect a single host from attacks), or *ii) active* (to cooperatively protect the nodes of an ad hoc cluster). In [22], a secure routing protocol based on AODV and IPv6 is proposed. It includes the SUCV mechanism (like in [25]) for non-repudiation and authentication, and it does not require the availability of a CA or KDC. RREQ and RREP packets have been extended by adding the RSA public key of the source node and the digital signature of the routing message. Upon receiving an RREQ message, each intermediate node authenticates the source node, by verifying the message integrity and by verifying the signature with the source node's public key. In this solution, the routing protocol and the IDS can be considered independent to each other. Each node (out of the route path) monitors the traffic activity within the radio range, to detect intrusions, determined as anomalous behavior of observed nodes.

Anomaly Intrusion Detection Based on Clustering a Data Stream

Sang-Hyun Oh¹, Jin-Suk Kang², Yung-Cheol Byun³,
Taikyeong T. Jeong⁴, and Won-Suk Lee¹

¹ Dept. of Computer Science, Yonsei Univ., Korea
{osh, leewo}@database.yonsei.ac.kr

² Dept. of Computer Eng., Kunsan National Univ., Korea
jskang01@kunsan.ac.kr

³ Dept. of Communication & Computer Eng., Cheju National Univ., Korea
ycb@cheju.ac.kr

⁴ Dept. of Electrical & Computer Engineering, University of Texas at Austin, USA
ttjeong@mail.utexas.edu

Abstract. In anomaly intrusion detection, how to model the normal behavior of activities performed by a user is an important issue. To extract the normal behavior as a profile, conventional data mining techniques are widely applied to a finite audit data set. However, these approaches can only model the static behavior of a user in the audit data set. This drawback can be overcome by viewing the continuous activities of a user as an audit data stream. This paper proposes a new clustering algorithm which continuously models a data stream. A set of features is used to represent the characteristics of an activity. For each feature, the clusters of feature values corresponding to activities observed so far in an audit data stream are identified by the proposed clustering algorithm for data streams. As a result, without maintaining any historical activity of a user physically, new activities of the user can be continuously reflected to the on-going result of clustering.

Keywords: Intrusion detection, Anomaly detection, Data mining, Clustering, Data stream.

1 Introduction

Due to the advance of computer and communication technology, the amount of damage caused by unexpected intrusions and other computer related crimes have been increased rapidly. Intrusion detection systems are usually classified as host-based or network-based in terms of target domain. A host-based system detects an intrusion for a single host by monitoring its system log. A network-based system detects an intrusion for a network by examining its on-going network packets. The methodology of intrusion detection is classified into a misuse detection model [1, 2] and an anomaly detection model [3, 4, 5, 6, 7]. In the misuse detection model, well-known attacks are manufactured to attack profiles. When an online user activity occurs, it is compared with attack profiles. If it is matched to one of attack profiles, it is regarded as an attack. The misuse detection model utilizes the well-known weaknesses of a target domain. Intrusion methods have, however, evolved into more sophisticated

forms and many new intrusion methods are being invented as well. As a result, individually handling well-known intrusion methods is no longer enough to preserve the security of a target domain. The anomaly detection model has been studied as a possible alternative for coping with this problem. In the anomaly detection model, the historical activities of a user are manufactured by a profiling method and then long-term profiles are generated. When an online user activity occurs, it is transformed to a short-term profile. Therefore, if the difference between the short-term profile and any one of the long-term profiles is large, the activity is regarded as an attack.

Recently, many data mining methods [8, 9, 10] for a data stream have been actively introduced. A data stream is an ordered sequence of objects o_1, \dots, o_n that must be accessed in order and that can be read only once or a small specified number of times. As a result, it is impossible to maintain all objects of a data stream in the main memory. Consequently, each object should be examined at most once to analyze a data stream. In addition, memory usage for data stream analysis should be confined finitely, although new objects are continuously generated in a data stream. So that it can be instantly utilized upon request, newly generated objects should be processed as quickly as possible in order to maintain an up-to-date analysis result of the data stream, so that it can be instantly utilized upon request. To satisfy these requirements, data stream processing sacrifices the correctness of its analysis results by allowing some errors.

Meanwhile, clustering methods [11, 12, 13, 14, 15] are suitable for modeling a large number of data objects as long as there exists a distance measure among them. This is because it is based on data similarity. Clustering is a process of partitioning a plain collection of data objects into meaningful groups called clusters. In other words, the purpose of clustering is to locate the groups of similar data objects which are defined by a given similarity measure. Consequently, the potential groups and structures of data objects in a data set can be identified.

In this paper, we propose an anomaly detection method based on clustering a data stream. In most conventional clustering methods on data streams, only a given number of clusters are identified. However, since the number of clusters in a data stream is unknown, their quality can be poor. On the other hand, in the proposed method, a cluster can be split into two clusters or two clusters can be merged into one cluster with respect to the distribution of objects occurring in a data stream. Therefore, clusters can be more effectively identified. In the proposed method, the various statistics of the objects in terms of identified clusters are modeled as a profile in order to improve the performance of anomaly detection. Whenever a new activity is performed, the profile is updated instantly. In addition, to evaluate the proposed method, synthetic data sets and 1998 DARPA data sets [16] are used.

This paper is organized as follows. In Section 2, related works are presented. Section 3 describes a method of clustering a generated data stream. In addition, it also presents how to continuously update the profile of the user. In Section 4, based on the profile of the user, an anomaly detection method is introduced. In Section 5, the simulation results of the proposed anomaly detection method are comparatively analyzed. Finally, this paper is concluded in Section 6.

2 Related Work

2.1 Intrusion Detection Methods

Typical anomaly detection models include statistical analysis [3, 4, 5], predictive pattern generation [6] and data mining method [7]. The statistical analysis maintains the historical activities of a user as a statistical profile. For a set of activities, the rate of inconsistency with the profile is represented as its anomaly rate. The typical systems of statistical analysis are IDES [3], NIDES [4], and EMERALD [5] developed in SRI. NIDES which is the improved version of IDES utilizes a statistical technique for anomaly detection as well as a rule-based technique for misuse detection. For anomaly detection, NIDES models the historical behavior of a user in terms of various features and generates a long-term profile containing a statistical summary for each feature. For detecting an anomaly, the information of the new activities of the user is summarized into a short-term profile, and then it is compared with the long-term profile of the user. If the difference between two profiles is large enough, the new activities are considered as anomalous behavior. In the EMERALD system that is similar to NIDES, the target of intrusion detection is extended from a single host to network environment. In the predictive pattern generation technique, it is assumed that the sequence of events follows a discernible pattern. This approach uses inductively generated time-based rules that characterize the patterns of normal user behavior. The rules are dynamically modified during the learning phase and only interesting rules remain in the system eventually. Therefore, an anomaly is detected if the observed sequence of events matches the left hand side of a rule but the subsequent events deviate significantly from those events predicted by the rule.

Predictive pattern generation [6] is a technique of anomaly detection that is based on the hypothesis that sequences of events are not random but follow a discernible pattern. These results in better intrusion detection because it takes into account the interrelationships and ordering among events. The approach of time-based inductive generation uses time-based rules that characterize the normal behavior patterns of users. The rules, generated inductively, are modified dynamically during the learning phase and only “*good*” rules, i.e., rules with high prediction accuracy and a high level of confidence remain in the system. A rule has high accuracy of prediction if it is correct most of the time, and it has a high level of confidence if it can be successfully applied many times in observed data.

For a network based anomaly detection system, agent-based intrusion detection methods [7] are developed. JAM (Java Agent for Meta Learning) [7] uses frequent-episode mining to generate the normal usage patterns of a specific node in a network. These patterns are used to build a base-classifier that determines the abnormality of the network node. In order to guarantee correct classification, a sufficient amount of normal and abnormal log data should be gathered for the learning phase of a classifier. A set of base-classifiers can be used to build a meta-classifier. Since each base-classifier monitors a different node of a network, an intrusion for the network can be detected by a meta-classifier combining the results of its base-classifiers. However, due to the nature of frequent episode mining, numeric data such as the size of a network packet may be modeled inaccurately. This is because it should be quantized to one of predefined ranges in order to represent it as a categorical data item.

2.2 Clustering Methods

Clustering techniques are categorized into several different methods: partitioning, hierarchical clustering, density-based clustering and grid-based clustering. Partitioning algorithm [11] divide the data space of a data set into mutually disjoint regions called clusters. In the hierarchical clustering algorithms [12, 13, 16], a pair of initial clusters are successively merged until the predefined number of clusters is left. Unlike other methods, the density-based clustering method regards a cluster as a region in a data space with the proper density of data elements. Typical density-based clustering methods are DBSCAN [14] and CLIQUE [15].

Clustering on a data stream is categorized by k-means/k-median and grid-based methods. In order to identify the clusters of objects occurring in a data stream, a k-median algorithm is proposed [8]. It regards a data stream as a sequence of stream chunks. A stream chunk is a set of consecutive objects occurring in a data stream. Whenever a new stream chunk containing a set of newly generated objects is formed, the LSEARCH routine which is an $O(1)$ -approximate k-median algorithm is performed to select k objects from objects of the stream chunk as the local centers of the chunk. The algorithm confines its memory space to holding the local centers of a fixed number of previous stream chunks. Therefore, if retaining ik centers is impossible at the i^{th} stream chunk, the LSEARCH routine is performed again to cluster the weighted ik local centers to retain k centers, which are the up-to-date centers of the k clusters for all the data elements generated so far in the data stream.

In the grid-based method [9], the data space of the feature is partitioned into a set of mutually exclusive equal-size *initial cells*. As a new user activity is performed continuously, each initial cell monitors the distribution statistics of its corresponding feature values within its range. When the transaction support of an initial cell g becomes dense enough, the range of the cell is dynamically divided into two mutually exclusive smaller cells, called *intermediate cells*. In addition, the distribution statistics of the cell g are used to estimate those of each divided cell. Similarly, when an intermediate cell itself becomes dense, it is partitioned by the same way. Eventually, a dense region of each initial cell is recursively partitioned until it becomes the smallest cell called *a unit cell*. A cluster is a group of adjacent dense unit cells. Each cluster represents the frequent range of the activities of the user with respect to the feature.

3 Clustering Streaming Data

When multiple dimensions are associated with each object in a data stream, multidimensional clustering can be used to find a set of multidimensional clusters in a data stream. However, the number of dimensions i.e. features can be large in anomaly detection and each object in a data stream is not necessarily related to all the dimensions of the data stream. In other words, each object maintains a set of values for only related dimensions. Therefore, in the proposed method, anomaly detection is performed by using a one-dimensional clustering method.

In most conventional clustering methods, the number of clusters is given in advance. However, since the number of clusters in a data stream is unknown, inaccurate clusters may be identified. Unlike conventional methods, in the proposed

method, a set of clusters are dynamically identified with respect to the distribution of objects occurring in a data stream. For the purpose, each object is considered as a cluster until the number of objects occurring in a data stream is the same as a given *initial_cluster_number*. In the proposed method, a center point represents each cluster. Therefore, whenever a new object occurs, the object is inserted into a cluster whose center is closest to the object. As time goes by, the range of each cluster can become too large and several clusters can be very close to one another. Therefore, in order to maintain the quality of the clusters identified from a data stream, any cluster whose range is too large should be split into more than two clusters and very close clusters should be merged into one cluster.

A data stream is represented by $S = \{o_1, o_2, \dots, o_h\}$ and an object o_i is represented by n -dimensional vector i.e., $o_i = (o_i^1, o_i^2, \dots, o_i^n)$. Therefore, a stream data projected to the k^{th} dimension from the data stream S is represented by $S^k = \{o_1^k, o_2^k, \dots, o_h^k\}$. Let X^k denote a set of clusters identified from S^k . In order to effectively maintain the quality of the clusters, each cluster has a set of grid-cells. The interval of a grid-cell is defined by a non-overlapped and equal-sized unit. In order to represent a unit, only one value, a *unit identifier*, is used in this paper. For instance, an object o^k is transformed to a unit identifier $I^k (I^k = \lceil o^k / \rho^k \rceil)$ where ρ^k represents an interval size for the k^{th} dimension. A grid-cell g^k contains the following information: the unit identifier I^k , linear sum gl^k , square sum gs^k and total number gn^k of objects contained in the interval of g^k i.e., $g^k = (I^k, gl^k, gs^k, gn^k)$. The properties of a cluster $C^k \in X^k$ are represented by Definition 1.

Definition 1. Cluster Properties

Given a cluster C^k of similar data objects for the k^{th} dimension, the properties of a cluster C^k are represented by a tuple $C^k(\delta^k, \mu^k, SS^k, gSet^k)$.

where, δ^k : The density of the cluster C^k is represented by δ^k . It is set to the total number of objects in the cluster C^k , and μ^k : The central value of the cluster C^k is represented by μ^k . It is calculated by the average of objects in the cluster i.e.,

$$\mu^k = \frac{1}{\delta^k} \cdot \sum_{j=1}^r o_j^k .$$

At the same time, SS^k : The square sum of objects contained in C^k

is represented by SS^k . It is calculated by the square sum of objects in C^k i.e.,

$$SS^k = \sum_{j=1}^r (o_j^k)^2 .$$

Furthermore, σ^k : The standard deviation of objects in C^k is

represented by σ^k . It is calculated by $\sigma^k = \sqrt{SS^k / \delta^k - (\mu^k)^2}$. Finally, $gSet^k$: The grid-cell set of objects contained in C^k is represented by $gSet^k$.

When a new object o^k occurs from a data stream S^k , a cluster whose center is closest to the object should be selected from X^k in order to insert the object o^k into the cluster. The updated properties of the cluster C^k are calculated as follows:

$$C^k \left(\delta^k + 1, \frac{\mu^k \cdot \delta^k + o^k}{\delta^k + 1}, SS^k + (o^k)^2, gSet^k \right)$$

To update the grid-cell set $gSet^k$ consider the only grid-cell whose interval contains the object o^k . In other words, when the unit identifier of a grid-cell g^k is same as the unit identifier of the object o^k , the properties of the grid-cell g^k can be updated. Let $\overline{gl^k}$, $\overline{gs^k}$ and $\overline{gn^k}$ denote the old properties of g^k and let gl^k , gs^k and gn^k denote the new properties of g^k , respectively. Ultimately, for a grid-cell $g^k \in gSet^k$ whose unit identifier I^k equals to $\lceil o^k/\rho^k \rceil$, the new properties of g^k are calculated as follows:

$$g^k = (gl^k + o^k, gs^k + (o^k)^2, gn^k + 1)$$

When many objects occur between two adjacent clusters, they are very close each other. If the standard deviation of all the objects in them becomes smaller than or equal to a user-defined threshold *minimum deviation*, they are merged into one cluster. For two adjacent clusters C_1^k and C_2^k contained in X^k , let σ denote the standard deviation of all objects in the two clusters and it is calculated as follows:

$$\sigma = \sqrt{\frac{SS_1^k + SS_2^k}{\delta_1^k + \delta_2^k} - \left(\frac{\mu_1^k \cdot \delta_1^k + \mu_2^k \cdot \delta_2^k}{\delta_1^k + \delta_2^k}\right)^2}$$

If *minimum deviation*, the two clusters are merged into one cluster C^k . The properties of the cluster C^k is updated as follows:

$$C^k \left(\delta_1^k + \delta_2^k, \frac{\mu_1^k \cdot \delta_1^k + \mu_2^k \cdot \delta_2^k}{\delta_1^k + \delta_2^k}, SS_1^k + SS_2^k, gSet_1^k \cup gSet_2^k \right)$$

In the above equations, the center of the cluster C^k is set to the weighted average of the centers of the two clusters C_1^k and C_2^k with respect to their densities δ_1^k and δ_2^k . Also, the square sum of the cluster C^k is set to the sum of SS_1^k and SS_2^k . When the cluster C_1^k and C_2^k are merged to the cluster C^k , the grid-cell set $gSet^k$ of C^k can be obtained by the union of their grid-cell sets $gSet_1^k$ and $gSet_2^k$, i.e., $gSet^k = gSet_1^k \cup gSet_2^k$. This is because the grid-cells of two clusters are not overlapped according to the definition of a grid-cell.

Meanwhile, as the number of objects occurring from a data stream becomes larger, the standard deviation of objects in each cluster becomes higher. In other words, the quality of each cluster can be low. Therefore, to maintain the quality of each cluster, the cluster is split into two clusters with respect to *minimum deviation* as follows. For a cluster $C^k \in X^k$, let $gSet^k$ be $\{g^k_1, g^k_2, \dots, g^k_p, \dots, g^k_q\}$. If $\sigma^k > \text{minimum_deviation}$, then the cluster C^k is split into two clusters C_1^k and C_2^k . Let T_i^k denote a set of objects contained in $g^k_i \in gSet^k$ and let r_p^k denote the total number of objects contained in

$$\bigcup_{i=1}^p T_i^k \text{ i.e., } r_p^k = \sum_{i=1}^p gn_i^k. \text{ If } r_{p-1}^k < \delta^k/2 \leq r_p^k < \delta^k, \text{ then the grid-cell sets of}$$

the two clusters are $gSet_1^k = \{g^k_1, g^k_2, \dots, g^k_p\}$ and $gSet_2^k = \{g^k_{p+1}, \dots, g^k_q\}$. As a result, the properties of C_1^k and C_2^k can be obtained as follows:

$$C_1^k \left(\sum_{i=1}^p g n_i^k, \frac{1}{r_p^k} \sum_{i=1}^p g l_i^p, \sum_{i=1}^p g s_i^p, \bigcup_{i=1}^p \{g_i^k\} \right)$$

$$C_2^k \left(\sum_{i=p+1}^q g n_i^k, \frac{1}{\delta^k - r_p^k} \sum_{i=p+1}^q g l_i^k, \sum_{i=p+1}^q g s_i^k, \bigcup_{i=p+1}^q \{g_i^k\} \right)$$

Algorithm 1 describes the process of clustering a data stream. In this algorithm, input parameters are a data stream S^k , a minimum deviation, and an initial cluster number. In Line 1, each object is generated as a cluster with respect to the initial cluster number. And then, when a new object occurs, the following processes are performed. In Lines 3~4, a cluster whose center is closest to a new object is selected from the cluster set and the properties of the cluster are updated. In Lines 5~10, when the standard deviation of all the objects in any two adjacent clusters is less than or equal to the minimum deviation, two clusters are merged. In Lines 11~15, when the standard deviation of any cluster becomes larger than the minimum deviation, the cluster is split into two clusters and then newly generated clusters are inserted into the cluster set X^k .

Algorithm 1. Clustering a data stream

Clustering (S^k , minimum_deviation, initial_cluster_number)

- 1: Generate initial clusters w.r.t initial_cluster_number and then insert them into X^k .
- 2: **foreach** $o \in S^k$ **do**
- 3: Select the closest cluster C^k in X^k from o .
- 4: Update the properties of the cluster C^k .
- 5: Select the most adjacent cluster $C^{k'}$ of the cluster C^k .
- 6: Calculate the standard deviation σ of objects in two clusters C^k and $C^{k'}$.
- 7: **if** $\sigma \leq$ minimum_deviation, **then**
- 8: Merge C^k and $C^{k'}$ into C^k .
- 9: Update the properties of the cluster C^k .
- 10: Delete $C^{k'}$ from X^k .
- 11: **if** $\sigma^k >$ minimum_deviation, **then**
- 12: Split C^k into two clusters.
- 13: Update the properties of the two clusters.
- 14: Insert the two clusters into X^k .
- 15: Delete C^k from X^k .

4 Anomaly Detection

For each feature, the on-going result of clustering is summarized in a profile, which is composed of the two properties of each cluster, a center and a standard deviation. An anomaly in a newly occurring object can be identified by comparing the new object with the current profile of each feature. For this purpose, as a new object occurs from a data stream, if the difference between the object and its closest cluster becomes large, this object is considered as an anomaly. The difference $diff(X^k, o^k)$ between an objects and its closest cluster C^k is defined as follows:

$$diff(X^k, o^k) = \frac{|\mu^k - o^k|}{\sigma^k} \quad (C^k \in X^k)$$

In the above equation, the distance between the cluster C^k and the object o^k should be divided by the standard deviation σ^k . This is because the common characteristics of the features should be normalized. As a result, when the number of features participating in anomaly detection is n , the overall abnormality of a new object can be calculated as follows:

$$abnormality(o) = \frac{1}{n} \cdot \sum_{i=1}^n diff(X^i, o^k)$$

In order to decide the rate of abnormal behavior in the new object o , a set of different abnormality levels can be defined relatively to the normal behavior of the historical activities. In this paper, two different abnormality levels (*green*, *red*) are considered in order to classify whether the activities of a new object are anomalous or not. The green level is safe while the red is warning. Let $\Upsilon(v, \lambda, \chi)$ denote the statistics of abnormalities until now. v , λ and χ are represented as the total number of objects occurring from a data stream S , the linear sum of their abnormalities and the square sum of their abnormalities, respectively. Based on the statistics Υ , the average Φ and its standard deviation Θ of abnormalities can be calculated as follows.

$$\Phi = \lambda/v, \Theta = \sqrt{\chi/v - \Phi^2}$$

The new object o is in

- *Green level*: if $0 \leq abnormality(o) \leq \Phi + \Theta \cdot \xi$
- *Red level*: if $\Phi + \Theta \cdot \xi < abnormality(o)$.

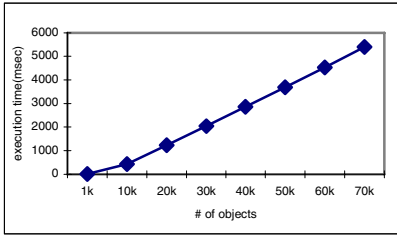
A detecting factor ξ is a user-defined parameter which determines how strictly the anomaly of a new object is classified. As it is decreased, a new object is more strictly examined. Given a set of normal object, its false alarm rate is represented by the ratio of the number of objects that are within the range of the red level over the total number of normal objects. Similarly, given a set of anomalous objects, its anomaly detection rate is represented by the ratio of the number of objects that are within the range of the red level over the total number of anomalous objects.

5 Experimental Results

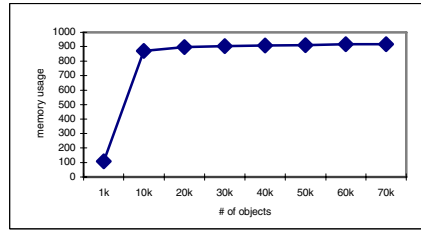
We present the results of experiments comparing the performance of LSEARCH and the proposed method. We conducted all experiments on a Pentium II with dual 350 MHz processors running LINUX 2.6.7. To demonstrate the performance of the proposed algorithm, synthetic data sets are generated as in Table 1. The data sets D20, D40 and D80 contain clusters which are explicitly separated and the number of clusters in each data set is 20, 40 and 80, respectively. The data set RAN contains randomly generated objects. In all experiments in this paper, the interval of grid-cell and the minimum deviation are set to 5 and 30, respectively.

Table 1. Synthetic data sets

Data sets	# of objects	# of clusters	Data size
D20	11,600	20	51 Kbytes
D40	23,200	40	109 Kbytes
D80	46,400	80	225 Kbytes
RAN	79,200	unknown	374 Kbytes



(a) Execution time



(b) Memory usage

Fig. 1. Performance of the proposed method

Figure 1-(a) illustrates the execution time and memory usage of the proposed method when the number of objects in the RAN data set is varied. In Figure 1, as the number of objects becomes larger, the execution time increases linearly. This is because the complexity of the proposed algorithm is $O(n)$. Figure 2-(b) illustrates the memory usage of the proposed method with respect to the number of objects occurring in the RAN data set. The memory usage is represented by the total number of grid-cells in clusters. In this experiment, the memory usage is saturated with about 900 when the number of objects is 1000.

Table 2. Average SSQ for each data set

	D20	D40	D80	RAN
Proposed Method	70.4	72.9	78.1	550.8
LSEARCH	70.0	13399.2	44858.6	13834.3

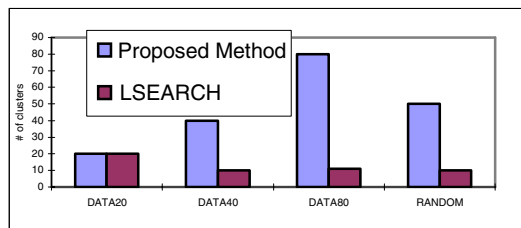
**Fig. 2.** The number of clusters

Table 1 illustrates the average SSQ's of the proposed method and LSEARCH which is popularly used for representing clustering quality. Average SSQ is the

average of squares of the distances to the cluster centers. In LSEARCH, the lower and upper bounds on the numbers of clusters to find are set to 10 and 80. For the data set D20, the average SSQ's of two methods are similar while those of LSEARCH are very higher than those of the proposed method for other data sets D40, D80 and RAN. It means that the proposed method finds clusters more correctly than LSEARCH. The reason of that result is shown in Figure 2. Figure 2 describes the number of clusters generated by the proposed method and LSEARCH. The proposed method finds correctly explicit clusters in data sets while LSEARCH do not.

In order to evaluate the performance of the proposed algorithm in a real world environment, we use DARPA log data sets collected in 1998 [17]. The feature values of the log data sets are extracted by BSM (Basic Security Module) [18] of Solaris 2.6. Among these signals, 84 signals are used as basic features in the experiments. In a log data set, an object is defined by the number of system calls occurring in a unix command on a host computer. We use two types of data sets for real world experiment: a programmer and a system administrator. A programmer writes a public domain C code via a "vi editor", compiles the C code (sometimes successfully), reads and sends mails, and executes unix commands. A system administrator runs privileged commands. In this experiment, the programmer is regarded as a target user for anomaly detection. To simulate the environment of each data stream, a data set is replicated multiple times and its transactions are looked up one by one in sequence.

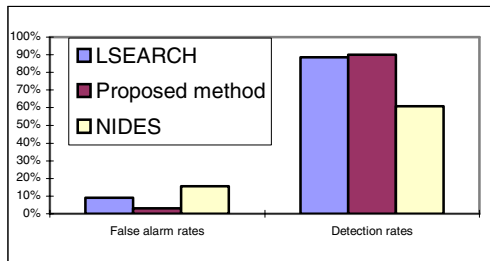


Fig. 3. Detection results

In Figure 3, the false alarm and detection rates in the proposed method are compared with those of the LSEARCH and NIDES. In this experiment, the value of the detecting factor ξ is set to 1.5. As shown in Figure 3, the false alarm rates of LSEARCH and NIDES are higher than that of the proposed method. Furthermore, the detection rate of NIDES is much lower than those of the proposed method and LSEARCH. As a result, the proposed method can detect an anomaly more effectively than LSEARCH and NIDES.

6 Conclusions and Future Work

This paper proposes an anomaly detection method that employs a clustering algorithm for a data stream. For each feature, its clusters can be effectively found upon without maintaining any object of the data stream physically. For the purpose, clusters are

dynamically generated by splitting a cluster into two clusters or merging two adjacent clusters into one cluster. As a result, the proposed method can find clusters more correctly than other conventional methods. For anomaly detection, new objects are continuously reflected to both the on-going result of clustering and the profile at the same time. Therefore, an anomaly can be detected easily without additional processes. In the future, the method of dynamically finding the interval size of a grid-cell and a minimum deviation is researched.

Acknowledgement

This work was supported by grant No.R01-2006-000-11223-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

- [1] B. Mukherjee, T.L. Heberlein, and K.N.Kevitt, "*Network Intrusion Detection*," IEEE Network, 8(3):26-41, May/June 1994.
- [2] R. Heady, G.Luger, A.Maccabe, and M.Servilla, "*The Architecture of a Network Level Intrusion Detection System*," Technical Report, Computer Science Department, University of New Mexico, August 1990.
- [3] H.S. Javitz, A. Valdes, "*The SRI IDES Statistical Anomaly Detector*," In Proc. of the 1991 IEEE Symposium on Research in Security and Privacy, May 1991.
- [4] H.S. Javitz and A. Valdes, The NIDES Statistical Component Description and Justification, Annual report, SRI International, 333 Ravenwood Avenue, Menlo Park, CA 94025, March 1994.
- [5] P.A. Porras and P.G. Neumann, "*EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances*," 20th NISSC, October 1997.
- [6] H.S. Teng, K. Chen, and S.C. Lu "*Security Audit Trail Analysis Using Inductively Generated Predictive Rules*," In Proceedings of the Sixth Conference on Artificial Intelligence Applications. pages 24-29, Piscataway, New Jersey, March 1990 IEEE.
- [7] S.J. Stolfo, A.L. Prodrmidis, S. Tselepis, W. Lee, D. Fan, P.K. Chan, "*JAM:Java agents for Meta-Learning over Distributed Databases*," .Proc. KDD-97 and AAAI97 Work. on AI Methods in Fraud and Risk Management), 1997.
- [8] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "*Clustering data streams: Theory and practice*," IEEE Trans. Knowl. Data Eng 15, 3 (2003), 515--528.
- [9] Nam Hun Park and Won Suk Lee, "*Statistical grid-based clustering over data streams*," SIGMOD Record, 33(1), pp. 32-37, 2004.
- [10] J.H. Chang and W. S. Lee, "*estWin: adaptively monitoring the recent change of frequent itemsets over online data streams*," CIKM 2003, pp. 536-539.
- [11] MacQueen, J., "*Some Methods for Classification and Analysis of Multivariate Observations*," Proc. 5th Berkeley Symp., 1967, Pages 281-297.
- [12] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, "*Birch: An Efficient data clustering method for very large databases*," Proceedings for the ACM SIGMOD Conference on Management of Data, Montreal, Canada, June 1996.
- [13] S. Guha, R. Rastogi and K. Shim, "*CURE: An Efficient Clustering Algorithm for Large Databases*," ACM SIGMOD International Conference on Management of Data, Seattle, Washington, 1998.

- [14] M. Ester, H.-P. Kriegel, J. Sander, X. Xu: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. 2nd int. Conf. on Knowledge Discovery and Data Mining (**KDD '96**), Portland, Oregon, 1996, AAAI Press, 1996.
- [15] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Seattle, Washington, June 1998.
- [16] T. Jeong and A. Ambler, "Power efficiency system for flight application (PESFA) mission: Low power dissipation in digital circuit design for flight application/space communications," *IEEE Tran. on Aerospace and Electronics Systems*, vol 42, 2006
- [17] <http://www.ll.mit.edu/IST/ideval/index.html>
- [18] Sun Microsystems. SunShield Basic Security Module Guid.

Robust Reactions to Potential Day-Zero Worms Through Cooperation and Validation

K. Anagnostakis¹, S. Ioannidis², A.D. Keromytis³, and M.B. Greenwald⁴

¹ Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613

² Computer Science Department, Stevens Institute of Technology, Hoboken, NJ 07030, USA

³ Department of Computer Science, Columbia University, 1214 Amsterdam Ave. New York, NY 10027, USA

⁴ Bell Labs, Lucent Technologies, Inc., 600-700 Mountain Ave., Murray Hill, NJ 07974, USA

Abstract. *Cooperative* defensive systems communicate and cooperate in their *response* to worm attacks, but determine the presence of a worm attack solely on local information. *Distributed* worm detection and immunization systems track suspicious behavior at multiple cooperating nodes to determine *whether* a worm attack is in progress. Earlier work has shown that cooperative systems can respond quickly to day-zero worms, while distributed defensive systems allow detectors to be more conservative (i.e. paranoid) about potential attacks because they manage false alarms efficiently.

In this paper we begin a preliminary investigation into the complex tradeoffs in such systems between communication costs, computation overhead, accuracy of the local tests, estimation of viral virulence, and the fraction of the network infected before the attack crests. We evaluate the effectiveness of different system configurations in various simulations. Our experiments show that distributed algorithms are better able to balance effectiveness against viruses with reduced cost in computation and communication when faced with false alarms. Furthermore, cooperative, distributed systems seem more robust against malicious participants in the immunization system than earlier cooperative but non-distributed approaches.

1 Introduction

Increasing innovation among attackers, the increasing penetration of broadband Internet service and persistent vulnerabilities in host software systems have led to new classes of rapid and scalable mechanized attacks on the information infrastructure. Leveling the playing field requires scalable, automated responses to malicious code that can react in the short propagation windows evident with network worms such as Slammer [1]. Traditional approaches have relied on signatures, manual containment and quarantine (e.g., [12]), and while tools are improving, reliance on identifying signatures and other improvements in detection processes is by itself insufficient. What is needed to complete the defensive

technology portfolio is a scalable, distributed, adaptive response mechanism, based on cooperative behavior amongst a set of responding nodes. Since naïve cooperative behaviors might introduce new risks, including fragility in the face of poor or maliciously-generated information, particular attention must be paid to robustness in the cooperative strategy.

The problem of detecting, quarantining and recovering from day-zero viruses¹ is made easier if local detectors are allowed more room for error. If we err on the side of allowing false alarms, then detectors can be cautious (paranoid!) and conservatively flag anything that looks suspicious, and depend on cooperative corroboration to determine whether the attack is real or not. For this strategy to be effective, though, requires the entire anti-virus system to handle false alarms quickly and cheaply and still respond rapidly to real virus attacks.

Handling individual false alarms is not sufficient, however; by allowing more false alarms we increase the probability that the system will be called upon to manage multiple simultaneous *potential* viral attacks. Simultaneous attacks complicate the anti-virus response because increasing the defense against one virus involves either decreasing the defense against another virus or incurring higher costs (if the system can afford any further anti-virus costs). Simultaneous attacks may occur because of multiple day-zero viruses [10], or because early-stage false-alarms are not yet distinguishable from real virus attacks, or because of the resurgence of old viruses. Old viruses are still potentially virulent, since a measurable fraction of hosts do not upgrade or patch to eliminate security bugs, as the persistence of Code Red and other worms has demonstrated. A good analysis of the persistence of Blaster is given in [4], which shows that tens of thousands of instances of the virus remain active a full year after the initial outbreak. More surprisingly, 73%–85% of infected class-C subnets were not detected as infected during the first Blaster outbreak.

These observations expose several significant problems that must be dealt with. Any node that responds to a potential virus carries a cost: a node has finite resources and therefore can only engage a limited number of viruses at a time. Deciding to counter one virus entails ignoring some other virus. In the absence of cost, the best response to a potential virus attack is to flood the network as rapidly as possible, causing as many cooperating agents to respond at once. The main question is simply whether the response is quick enough to stifle the virus. In the presence of a cost model, however, we still need to respond quickly, but no more quickly than necessary. A false alarm, whether malicious or unintended, can trigger a DoS attack by the response mechanism itself.

In this paper, we investigate tradeoffs in global, distributed response mechanisms that must respond quickly to real viruses and do not over-react to false alarms. These systems should be efficient in terms of bandwidth and global

¹ In this paper we use the term “virus”, in an extremely broad manner, to refer to any epidemic-like attack communicated over the network. We use this term, regardless, whether the virus is an active worm that attacks a system even without the unwitting involvement of a legitimate user or a passive virus, that is embedded in a document or email, that requires (unintentional) user assistance to become active.

computation. Moreover, the response mechanism must be robust against malicious agents spreading false information and be able to manage its resources even when many distinct viruses are active at any time. This approach is orthogonal to, and can augment, any proposals for the detection of and recovery from day-zero viruses. It has the added advantage of also performing well in the face of false alarms resulting from malicious behavior or failed detectors.

We focus here on an algorithm called COVERAGE, whose core ideas were originally introduced in [2]. This algorithm takes into consideration shared information about the observed rate of infection for each virus, verifying that new reports are compatible with a node's own empirical observations, and determines (probabilistically) which viruses to respond to. We evaluate its effectiveness through large-scale simulation. We discuss also, although to a lesser extent, tradeoffs in a similar cooperative (but non-distributed) approach, called NRL03, described in [13], which differentiates between slow and fast-spreading viruses.

In our previous work, we determined that our basic cooperative and distributed approach was effective, but only in the sense of measuring the ability of the two approaches to detect and respond to worms of different infection rates, as well as their resistance to malicious nodes that spread misinformation. Here, we offer refined algorithms over the earlier COVERAGE work, and, using a more detailed model we begin to examine the three-way tradeoff between communication costs, computation overhead, and the percentage of the network that gets infected before the reaction mechanism manages to limit the worm propagation. When evaluating alternative approaches in this model, we determine that COVERAGE always has much lower scanning costs; whether in the cases of slowly-propagating worms, fast worms, or in the presence of malicious nodes injecting false alarms. COVERAGE has significantly lower communication cost when dealing with false alarms. Furthermore, the distributed approaches (both COVERAGE and our refined version) can trade off higher communication cost to to detect and react to slow-propagating worms more quickly than the purely cooperative NRL03. Finally, both NRL03 and the COVERAGE variants can use increased communication costs to perform better against fast worms, but the distributed approaches require much higher communication costs than NRL03 — perhaps an unavoidable consequence of their robustness against false alarms.

2 System Model

In this section, we describe our model of how viruses, switches/routers, hosts and our detection mechanism behave.

Modeling Viruses. We use a fairly simple model to describe the behavior of potential attackers (viruses) that we consider in our work. After infecting a node, a virus attempts to infect other nodes; it may attempt to only infect a (small) fixed number of other nodes, or exhibit a greedier behavior. For our purposes, the distinction between the two types is simply in the probability of detection of a probe or attack by a detector. A virus may exhibit high locality of infection

(*i.e.*, probing and attacking nodes based on network-topological criteria, such as “adjacent” IP addresses), or could use a random (or seemingly random) targeting mechanism, *e.g.*, using a large hit-list, or some pseudo-random sequence for picking the next address to attack. We expect that viruses that exhibit high locality are more difficult to detect using an Internet-wide distributed detection mechanism, but easier to do so on a local basis. We completely characterize a virus by the rate at which it attempts to infect other nodes and by the fraction of local attempts it makes. All attacks on susceptible nodes are successful, and in our simulation a virus never attempts to attack a non-existent node. As a result, our simulated viruses are more virulent than equally aggressive viruses in the real world. We make no assumptions about the infection vector: although perhaps the more “interesting” cases are those where the virus is able to automatically subvert a machine or application, our model does not preclude human interaction in the infection process (*e.g.*, mail viruses as attachments).

Furthermore, we only assume that, once detected, there is some detection and/or response “module” associated with each virus — we do not investigate its details: the mechanism may be as simple as a content filter. There is some cost (in terms of CPU, memory, impact on legitimate communications, *etc.*) associated with each of these modules, which requires the prioritization of the various threats (viruses) in terms of allocating resources for detection and response.

Detection of Zero Day Worms. Although our algorithm is orthogonal to and agnostic about the method(s) with which new (zero day) worms are detected, we briefly discuss different techniques and how they may interact with COVERAGE. A zero day worm detector consists of roughly three components. First, we must detect anomalous behavior. The behavior may range from specific activities (*e.g.* port scans, system/application crashes, incorrect password attempts) to statistical changes (*e.g.* increased network traffic, slow response time, variation in system call signatures, number of TCP connections in TIME-WAIT). Second, the transmission vector must be identified (finding a set of network packets whose arrival seems to herald the onset of the anomalous behavior). Third, a detectable “signature” of the traffic must be generated so that hosts can scan for, and filter out, the potentially offending traffic. It is important to note that a “signature” in our model is not necessarily simply a pattern of bits to match inside a packet — it can be any profile that detects anomalous behavior, ranging from packet inspection to longer term multi-packet behavior.

Perhaps the most promising approach is that of monitoring the number of packets aimed at the unused portion of an organization’s address space, as was suggested in [6]. In that work, it was shown that with as few as 4 such probes, it is possible to infer the existence of a new worm aimed at a previously untargeted service/port. A similar approach is proposed in [19], where sudden changes in the traffic statistics maintained on a per source IP address and per destination port number indicate a high-visibility event, such as a scanning worm. Similar works have proposed measuring the entropy of traffic (*e.g.*, in terms of distinct source IP addresses seen) as an indication of unusual activity. These mechanisms act as early warnings, alerting administrators and perhaps automatically reconfiguring

a firewall to assume a more defensive posture. However, without corroboration with outside sources (*e.g.*, through COVERAGE) they can be manipulated by an attacker to generate false positive reports. It is also worth noting that these mechanisms can only give a rough fingerprint of a new worm's activity, such as the targeted service/port—thus, they can be fairly accurate about the presence of an attack, but inaccurate about mapping specific packets to the attack, as would be the case with a worm targeting a protocol such as HTTP.

A second, more accurate but also more expensive (computationally, as well as in terms of necessary infrastructure) mechanism for detecting worms is through use of properly instrumented honeypots or virtual machines, as is done in [16,9], or through payload analysis [8,17] that can yield a potential worm signature. Finally, anomaly detection techniques, such as those proposed in [5], can indicate the presence of packet payloads that do not conform to the typical contents of packets for a particular service (*e.g.*, binary content containing a buffer overflow payload uploaded to a web server).

These mechanisms identify different points in the zero-day worm detection space, trading off between the likelihood of false positives, the time needed to collect enough evidence before raising an alarm, and the expense of testing whether an alarm should go off.

These observations are taken into consideration by the COVERAGE algorithm to balance the cost of detection (*e.g.*, coordination, scanning as well as collateral damage that may be caused by false alarms) and the ability to respond effectively to virus attacks.

Network Topology. Our simulation topology is dictated by our assumptions about the vulnerabilities and capabilities of network nodes with respect to virus attacks. We assume that, as a general rule, routers/switches are less likely to be infected by a virus, and thus that only hosts are susceptible to infection.

Here, we assume that the only nodes in our system capable of scanning packet sequences for potential viruses are end-hosts or last-hop routers. While considerable advantage can be gained by exploiting the great levels of traffic aggregation seen in routers closer to the network core, it is unlikely that such nodes can actively scan for viruses without significantly affecting their performance.

Thus, our model of the network topology consists entirely of a collection of *subnets* (LANs) containing a number of *hosts*. Each subnet connects to the global network through a single *router*. All routers are connected together in a single cloud where each router can address and forward packets to each other directly. End-hosts can only see their traffic, while routers can inspect all traffic to or from their associated LAN. It is likely that some organizations contain multiple subnets that frequently communicate among themselves. Therefore we collect together several subnets into a *domain*. A domain captures particular communication patterns but has no structural impact on the topology for simulation.

State of Nodes. A node in our environment can be in one of three states with respect to a virus: *susceptible*, *protected*, or *immune*. A susceptible node can be either infected or uninfected. Susceptible nodes will become infected if subjected

to an attack. Protected nodes may be infected or uninfected, but only if the detection module does not have the ability to detect and disinfect an infected machine. A protected node will not become infected as long as the protection mechanism (typically, a module that screens packets or email) is in place. An immune node does not have the vulnerability exploited by the virus.

Operations. A COVERAGE agent can monitor traffic and, for each virus, it can either ignore the virus or perform one or more of the following operations: collect and exchange information about a virus, *scan* for the presence of a virus (actually, scan for the presence of patterns of network traffic used as a “signature” for that virus), or *filter* viruses (by dropping one or more packets that are part of a virus signature). We assume that there is a cost inherent in checking for virus signatures. That is, a node cannot be actively “on the lookout” for an arbitrary number of viruses without adversely affecting its performance. (Some experimental measurements of such real-world limits are given in [3]). Edge-routers are more likely to be constrained by high packet rates, and therefore limited in the amount of scanning they can perform. Hosts can afford to scan for more viruses without interfering with their (lower) packet rate, but, on the other hand, have work other than packet forwarding to perform. In either case there is an upper bound on the number of viruses a node can scan for.

We assume that nodes periodically exchange information about viral infections. Although the per-virus cost of such an exchange is low, we assume that the number of known *plus potential day-zero* viruses exceeds the amount of information that can be reasonably exchanged at any given time. Thus, actively exchanging information about a virus incurs a cost, albeit lower than scanning.

Routers can additionally scan for suspicious behavior on all traffic to or from their LAN (and drop when necessary). We further assume that if a router detects a rampant viral infection for a virus that has an associated disinfectant component, the router can invoke a disinfection operation (perhaps alerting an administrator) on all the nodes in its LAN.

Model of Anti-virus Epidemic. Each node participating in the anti-virus response must make certain decisions: (a) the rate at which it polls other local nodes for virus information, (b) the rate at which it polls other remote nodes, chosen at random, for virus information, (c) whether for each virus to collect information about it, (d), whether to include that information in virus exchange packets, and (e) whether to scan for the virus (collecting the results of those scans as part of the local information for that virus).

3 Cooperative Virus Response

COVERAGE tries to balance the cost of scanning and filtering packets for a specific virus against the benefit of detecting, other, real viruses in several ways. First, COVERAGE models the virulence of viruses and ranks them in virulence order. With probability proportional to their virulence, COVERAGE decides

in rank order whether to actively scan for the virus or not. It stops making decisions, and scans no more viruses, once the scanning schedule consumes the entire scanning budget available. Second, each COVERAGE agent exchanges information about the state of a virus with other cooperating agents in order to construct a model of the virus and determine whether incoming reports are empirically consistent with the observed state of the network. Third, COVERAGE agents determine their polling rate to maximize the probability of seeing enough viruses to confirm the current local estimate of the virus state, while reducing the probability that communication will add no new knowledge to either of the participants. We now describe the algorithm in more detail.

3.1 COVERAGE Algorithm

Agent communication. Each COVERAGE agent polls other agents, selected randomly. Assuming that only a small fraction of the nodes are reporting false information, a randomly selected node is more likely to be trustworthy than a node that actively contacts us — a small number of malicious nodes may try to flood the rest of the network. At each poll, the sender reads the response and updates its local state variables to track the operation of the cooperative response mechanism and the status of the network in terms of observed attacks.

First, it records whether the remote agent is actively scanning. This allows the agent to estimate the fraction of agents in the network that are actively scanning for a particular virus. Second, it updates estimates of possible infections *e.g.*, the fraction of infected nodes for each virus. We distinguish two types of estimates: direct and remote. Direct estimates are updated based on whether each remote agent has directly observed an attack (either to itself or, if a router, to a node in its LAN). Remote estimates are updated based on the fraction of infected nodes as estimated by the remote agent (the “direct” estimates of the remote agent). Direct measurements performed by the local node are absolutely trustworthy — there is no issue of false positives. The direct measurements of agents that we poll (which become our remote estimates) are next in trustworthiness. Remote estimates of agents who we poll are more suspect, and information reported by agents who contact us are the most suspicious of all. However, we can validate any information reported to us — if someone reports that a particular virus is attacking 25% of the Internet at the moment, then if we poll 20 agents at random, then with 80% probability we would expect to find that between 3 and 7 of those agents had directly seen an attack in the last measurement interval. Values outside that range would cast doubt on the remote estimate.

Finally, in this paper we ignore the details of how COVERAGE nodes authenticate themselves to each other. However, we note that even strong authentication is not sufficient for our system. If a COVERAGE agent is taken over by a malicious attacker, then the attacker can (presumably) still authenticate itself and discover which nodes are *not* scanning for a particular virus, and use that information when choosing targets. To defend against such a vulnerability in COVERAGE, we propose (but have not yet implemented or experimented with) a simple defense. When polling, the identity of the target agent is not important — just the fact that we

chose it randomly, and it did not choose us. And, while we are interested in the statistics of the sample as a whole, we need not link a particular set of direct measurements to a particular IP address. Consequently, each agent stores a randomly selected response from the last measurement interval (the local measurements are one of the candidates that may be selected), and returns that random selection in response to any COVERAGE poll, for the direct measurements and scan list only. (The cumulative counters are still stored and reported accurately). The poller still receives an accurate response — just perhaps from a different IP address than the one it polled, and perhaps slightly older than expected. This adds a level of indirection to the polling process.

Periodic updates. At regular intervals each COVERAGE agent updates its state based on the information received since the last update. To track the progress of the infection each COVERAGE agent maintains a smoothed history for each type of estimate (direct and remote), each as exponentially decaying averages with varying time constants, to approximate recent infection rate, past rate, and background rate.

Using these estimates, an agent can compute the fraction of nodes believed to be infected as well as the growth of the infection, assuming exponential growth².

If we assume that each infected node infects α nodes at each timestep, and that a fraction p^* of all nodes are infected at some start time t_0 , then at time t we expect the fraction of infected nodes to be $p^*(1 + \alpha)^{t-t_0}$. Consequently, if our direct samples at times t_0, t_1 , and t_2 report a fraction $p_d[t]$ nodes are infected³, we can estimate p_d^* and α_d for future growth by fitting

$$\begin{aligned} p_d[t_0] &= p_d^* \\ p_d[t_1] &= p_d^*(1 + \alpha_d)^{(t_1-t_0)} \\ p_d[t_2] &= p_d^*(1 + \alpha_d)^{(t_2-t_0)} \end{aligned}$$

Given estimates of p_d^* and α_d , we can calculate the *virulence*, v_d , of a virus as the estimated number of timesteps needed by the virus to infect the entire network.

Note that we independently calculate virulence for global and local growth, in order to identify attacks that are non-uniformly distributed throughout the network. Using the same method as above the agent also computes α_r , p_r^* and v_r based on the remote estimates.

Scanning/filtering. Given the estimates an agent can decide whether it needs to scan for a given virus. There is a basic, low level of scanning for every virus.

² We assume all growth is exponential for the purpose of deciding whether to trigger a reaction. We believe that linear growth worms can be detected by humans, and need not be countered by an automatic, distributed, algorithm. If our assumption is incorrect and growth is, in practice, sub-exponential then we recover naturally because we observe a decrease in α and gradually back-off as the predicted “virulence” of the virus drops.

³ In fact, we use the smoothed averages rather than instantaneous samples, and the details of the actual calculation are a bit more complicated, but are not relevant to the main point of this paper.

When a virus becomes active the scanning rate may increase. In the general case, the agent can sort viruses in order of their virulence v_d and decide whether to scan for each virus, in turn, stopping when the scanning budget is filled. (In our simulation, we only scan viruses whose v_d is below *threshold*.)

To maintain a basic, low level of scanning for every virus, every agent measures the fraction $f_{scanning}$ of nodes in the network that are actively scanning for a given virus based on information exchanged with other nodes. If this fraction is below a threshold f_{target} (around 2-5%) and the node has enough resources for scanning, it activates with probability $f_{target} - f_{scanning}$, and disables scanning in a similar way if too many nodes seem to be active. To avoid turning “blind” to certain worms because of a fraction of malicious nodes falsely reporting that they are actively scanning, nodes need to aim for $f_{target} + f_{malicious}$, where $f_{malicious}$ is the maximum tolerable fraction of malicious nodes. Although this increases scanning cost, nodes can trade-off this cost for higher communication costs.

An inactive agent, A , may also start scanning seemingly low-virulence viruses, if *enough other agents* claim the virus is virulent, and A finds that the fraction of scanning nodes is too low to detect virus activity in a single timestep at the current polling rate. The test is whether n (simply the fraction of agents that were polled and found to be scanning in the last interval) is less than twice the estimated fraction of infected hosts (*e.g.*, if $n < 2p_r^*$). Similarly, if the agent is active but $n > p_r^*$ then it decides to stop. The agent also stops scanning if α_r approaches 0. This ensures that the fraction of scanning agents is bounded if there is insignificant progress for a given infection or if the infection is small compared to the number of actively scanning agents. Such heuristics are essential for controlling the behavior of the algorithm, keeping the response mechanism “ahead” of the virus but also limiting the damage and cost when malicious agents spread false information.

A small number of agents need to be watching for each dormant virus. The number of active scanners monitoring a virus may be more than warranted by the level of virus activity. An agent detecting this will stop monitoring the virus. If the agent finds that it now has ample room within its scanning budget to consider another virus, it chooses another virus to monitor uniformly at random from the (large) virus database. The agent may choose a virus that almost no one else is scanning for — in which case it will stay on the scanning list for a long time, and be inspected by the agent as long as there are not too many virulent virii. If the new virus is dormant, but enough people are already looking at it, then the agent will drop it, and randomly choose another.

Polling rate. An agent communicates with agents within the same domain at a constant, high rate, as the cost of intra-domain communication is assumed to be very small. Inter-domain communication is generally more expensive; agents therefore need to adapt the rate of polling remote agents, avoiding excessive communication unless necessary for countering an attack. When there is no virus activity, agents poll at a pre-configured minimum rate (at least an order of magnitude lower than the rate for intra-domain communication). An agent periodically adapts the remote polling rate if v_r is less than a given threshold.

The new rate is set so that the agent polls $1/(p_r^*)^2$ remote agents in each update interval, unless this rate exceeds a pre-configured maximum rate. This is used to increase the polling rate when the remote estimate indicates that an attack is imminent (but not yet reflected in the direct estimate). If the more recent direct estimate $p_d[n]$ is non-zero, then the polling rate is increased so that at least a few samples can be collected in each update interval. Finally, if the estimated virus population p_r^* is small and the estimated virus growth rate is close to zero, the agent throttles back its remote polling rate to the minimum rate.

These adjustments are always performed on the polling side. We avoid changing the state or behavior of the polled agent to reduce the risks associated with malicious agents. Otherwise, they could spread misinformation and raise false alarms more effectively by increasing their own communication rate.

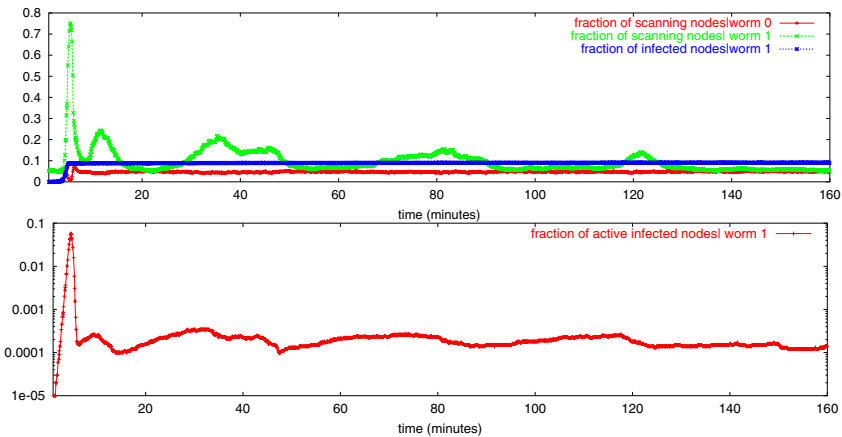


Fig. 1. Fractions of infected hosts and scanning nodes over time (top), and fraction of actively infected nodes (bottom)

3.2 COVERAGE Behavior

To give a rough sense of how the COVERAGE algorithms described above behave, Figure 1 displays a single example run of the COVERAGE algorithm against a single simulated virus called “worm 1”. We show the activity of the virus (the number of nodes that were ever infected in their lifetime) in (a), and the currently infected nodes in (b)), as well as the response of COVERAGE (both the number of agents scanning for “worm 1”, as well as the number of agents scanning for a dormant virus “worm 0”). (Section 4 describes how we approximate a heavy load on the COVERAGE agents by using a simulation parameter `threshold` — each agent is too busy to consider any virii unless they are likely to take over the entire network within `threshold` measurement intervals.)

One can see the initial stage of the infection and the response of the algorithm: the virus manages to infect roughly 10% of the hosts; cooperation between COVERAGE agents results in a rapid activation of filtering on roughly

75% of the network effectively eliminating the virus. Soon after stopping the attack, the COVERAGE agents on uninfected parts of the network deactivate scanning/filtering. However, as shown in Figure 1(b), a small number of hosts remains infected and undiscovered, resulting in another three episodes where COVERAGE agents are activated (each episode with a smaller fraction of agents activated) to defend against a secondary outbreak. Although a tiny fraction of infected nodes remains undiscovered, it does not cause any further harm and COVERAGE gives users time to patch up their systems. The scanning for dormant “worm 0” continues, except during the most virulent part of the outbreak, where the number dips as resources are marshaled to defend against “worm 1”.

4 Simulation Results

To simplify the analysis of COVERAGE and to meaningfully include the non-adaptive NRL03, we restrict the simulation to a single virus. We model the impact of multiple active viruses by assuming that each node is busy handling other viruses. To represent the load imposed by other viruses, we specify a threshold under which a virus will not have high enough priority to be scheduled in the scanning budget. If many viruses are active then the threshold will be a small number, such as 5 (recall that the virulence is a measure of how many measurement intervals it will take before the virus has covered the *entire* Internet). Unless the current virus is poised to conquer the entire net at its current rate of growth from its current coverage within `threshold` intervals, it will not have high enough priority to be scheduled in the scanning budget. We only consider cases where the net is already under heavy attack by other viruses, setting `threshold` equal to 5 and 20.

To better understand the performance of COVERAGE, we limit our simulation to a simple, relatively small network of 100,000 edge-routers, each connected to 8 hosts, with 50 edge-routers in each of 2,000 domains. We also consider the performance of our version of COVERAGE in relation to NRL03 [13], another cooperative algorithm, which makes different tradeoffs than COVERAGE. NRL03 uses cooperative peer-to-peer strategies to respond to large scale Internet worm attacks. The model involves a number of *friend* nodes, which work together by exchanging information to warn of suspicious worm-like network behavior. As in COVERAGE, a small fraction of nodes is assumed to be scanning for a given virus. When the virus is detected, the node broadcasts the alert to its friends. When a node receives such an alert, it increments an alert counter, and propagates the alert to its set of friends when this counter reaches a threshold.

For the COVERAGE algorithm, we set the local-domain polling interval to 1.8 seconds, the maximum and minimum remote polling intervals to 6 seconds and 1.8 seconds respectively. For both algorithms we assume that 4% of the edge-routers are permanently scanning for the virus.

Our analysis uses three metrics. First, we model the success of the attack by integrating the number of infected nodes over time. This is only relevant in

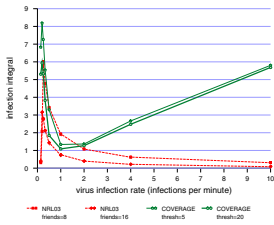


Fig. 2. Integral of infected hosts over time *vs.* virus infection rate

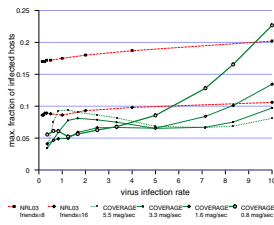


Fig. 3. Maximum fraction of infected hosts *vs.* virus infection when under attack

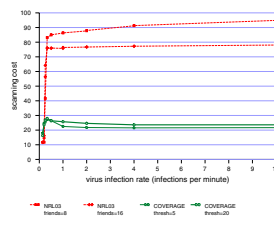


Fig. 4. Scanning activity *vs.* virus infection rate when under attack

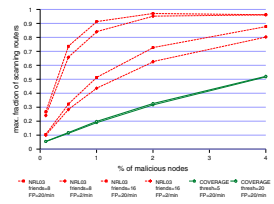


Fig. 5. Maximum fraction of false scanning routers *vs.* virus infection rate when under attack with malicious nodes

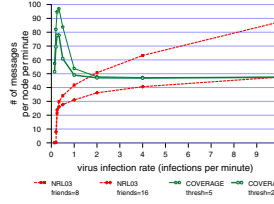


Fig. 6. Communication cost *vs.* virus infection rate when under attack

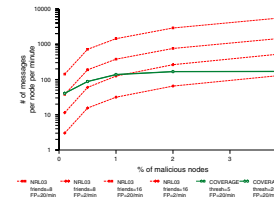


Fig. 7. The impact of malicious routers and false alarms on communication cost

the case of a real virus attack. Second, we consider the number of edge-routers actively scanning/filtering this virus. This is a measure of the computational overhead of the response mechanism. Our third metric, the total number of messages sent, measures the communication cost.

We measure the progress of infections of differing virulence and the success of the response mechanism as the integral over time of the fraction of infected nodes. The results for COVERAGE and NRL03 with different parameters are shown in Figure 2. (It may seem counter-intuitive that the more virulent viruses cover less of the network; however, recall that we are integrating over time and that the faster worms, while they spread faster are also detected and disinfect faster). COVERAGE reacts more slowly than NRL03 for fast worms — we model COVERAGE dealing with other viral outbreaks, but let NRL03 assume that this is the only virus in the Internet. Consequently, the virus takes a larger initial toehold in the network under COVERAGE, and is active slightly longer before being cleaned up. Because of this toehold, COVERAGE performs relatively worse than NRL for fast worms. On the other hand, it detects the slow worms before NRL03, and therefore does better. The slow response by COVERAGE in the case of fast worms has been a deliberate design choice in an attempt to make the algorithm robust against false information from malicious nodes. Figure 3 plots the high water mark of viral attacks as a function of the virus infection rate for different communication settings in COVERAGE. We can see that a moderate increase in communication rates in COVERAGE allows it to

stop the virus with a lower high water mark than NRL, but at the expense of more communication.⁴

Figures 4 and 5 show the fraction of nodes scanning for a virus as a function of the virulence of the virus and the fraction of malicious (or faulty) nodes. The figures for COVERAGE are more pessimistic than for NRL, because in NRL *every* edge-router is scanning for the virus (obviously impractical for a large set of viruses), and the graphs report only those nodes that are actively *filtering* the virus. The COVERAGE plots report the fraction of nodes that even scan for the virus at all. A smaller number (unreported here) are filtering for the virus. Nevertheless, a pessimistic (“worst-case”) results for COVERAGE have far lower scanning costs than optimistic (“best-case”) results for NRL03. COVERAGE manages to control the virus with a much smaller set of scanning nodes, and it similarly detects false alarms with fewer nodes triggered to scan or filter.

Figures 6 and 7 demonstrate that the communication costs for COVERAGE in the face of false alarms is much lower than for NRL03 — understandably because COVERAGE identifies the false alarms correctly. In the case of slow-growth worms, COVERAGE requires significantly more communication to convince co-operating peers that a virus attack *is* underway. However, this extra cost conveys a benefit: COVERAGE detects slow-growth worms long before NRL03 is able to. For fast worms, communication costs are generally comparable — NRL requires considerably more communication when Friends = 8, but it should be noted that NRL03 controls the infection more rapidly than COVERAGE in these cases. For COVERAGE to control fast worms as effectively as NRL03 would require even higher communication costs.

The impact of false alarms on detection performance (because nodes get confusing reports from malicious nodes about another virus) is illustrated more clearly in Figure 8. We see that the fraction of nodes that are left unprotected by COVERAGE grows almost linearly with the number of malicious nodes — roughly double the number of nodes are infected when 4% of the nodes are malicious compared to a system without any malicious nodes.

5 Related Work

Reference [15] shows that a distributed worm monitor can detect non-uniform scanning worms two to four times as fast as a centralized telescope, and that knowledge of the vulnerability density of the population can further improve detection time.

In [11], the authors coordinate the sharing of IDS alerts for detecting worm attacks and port scanning across administrative domains. Porras *et al.* [14] argue that hybrid defenses using complementary techniques (in their case, connection

⁴ The communication costs for COVERAGE scale with the virulence and number of active viruses, and are thus more scalable than NRL — still, we are investigating ways of conveying the necessary polling information more efficiently during quiet periods.

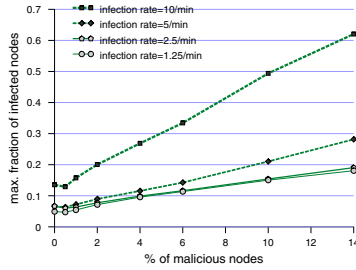


Fig. 8. Impact of false alarms and malicious nodes on detection performance

throttling at the domain gateway and a peer-based coordination mechanism), can be much more effective against worms.

Reference [18] proposes the use of “predator” viruses that spread in much the same way malicious viruses do but try to eliminate their designated “victim” viruses. The authors show that predators can be made to perform their tasks without flooding the network and consuming all available resources. However, designers of predators would have to find their own exploits (or safeguard exploits for future use), which is not an attractive proposition. Furthermore, many recent worms have been closing the hole they exploited, after infecting a machine.

DOMINO [20] is an overlay system for cooperative intrusion detection. The system is organized in two layers, with a small core of trusted nodes and a larger collection of nodes connected to the core. The experimental analysis demonstrates that a coordinated approach has the potential of providing early warning for large-scale attacks while reducing potential false alarms. Reference [21] describes an architecture and models for an early warning system, where the participating nodes/routers propagate alarm reports towards a centralized site for analysis. The question of how to respond to alerts is not addressed, and, similar to DOMINO, the use of a centralized collection and analysis facility is weak against worms attacking the early warning infrastructure.

The earliest work on cooperative response mechanisms is that of Nojiri *et al.* [13]. They present a cooperative response algorithm where edge-routers share attack reports a small set of other edge-routers. Edge-routers update their alert level based on the shared attack reports and decide whether to enable traffic filtering and blocking for a particular attack. Analysis by Kannan *et al* [7] has shown that cooperative response algorithms can improve containment, even when a minority of firewalls cooperate. That work, however promising, does not directly relate to our work. They are more concerned with a single fast virus — the analysis focuses on a single virus (consequently underplaying the cost of over-aggressive response), has a weaker model of “malicious” firewalls (malicious firewalls merely stay silent, but do not mislead through false alarms), and does not explore the benefits of allowing more latitude in generating false alarms.

6 Conclusions and Future Plans

We have described an algorithm, named COVERAGE, that allows cooperating agents to share information about the spread of malicious virus in the Internet and use this information for controlling the behavior of detection and filtering resources. The algorithm operates without fully trusting such information, so as to limit the damage of false alarms injected by malicious nodes. Our solution is based on the idea of carefully sampling of global state to validate claims made by individual participants. Simulation results confirm that this method is effective in limiting the damage of virus attacks, and that it is robust against attacks by malicious participants. When compared against a similar approach, the NRL03 algorithm [13], COVERAGE exhibits a lower cost in terms of scanning for worms due to its resource-aware approach. Furthermore, it has a lower communication cost in the presence of false alarms and fast worms, and can detect and react to slow-propagating worms better. However, it does not react as quickly to fast worms, and the price it pays for reacting to slow worms more quickly is higher communication overhead than NRL03 for slow worms.

Our plans for future work include reducing the communication cost of polling without measurably reducing the effectiveness of the mechanism, and examining in detail the case of multiple, simultaneously active viruses. We believe that we can tune the communication rate to adapt to the virulence of a virus, allowing COVERAGE to react to fast worms quickly (at the cost of increased communication overhead for very virulent worms). Due to space considerations, this paper simply assumes that many COVERAGE nodes are occupied by higher-virulence viruses, and that we must reserve processing cycles to deal with lower virulence viruses, too. Much work remains to be done in improving the actual choices each node makes of which set of viruses to monitor.

Acknowledgements. This work was supported in part by the National Science Foundation under grants ITR CNS-0426623, DUE-0417085 and CCR-0331584.

References

1. Cert Advisory CA-2003-04: MS-SQL Server Worm.
<http://www.cert.org/advisories/CA-2003-04.html>, January 2003.
2. K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, A. D. Keromytis, and D. Li. A Cooperative Immunization System for an Untrusting Internet. In *Proceedings of the 11th IEEE International Conference on Networking (ICON)*, pages 403–408, September/October 2003.
3. K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, and S. Miltchev. Open Packet Monitoring on FLAME: Safety, Performance and Applications. In *Proceedings of the 4th International Working Conference on Active Networks*, December 2002.
4. M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. The Blaster Worm: Then and Now. *IEEE Security & Privacy*, 3(4):26–31, July/August 2005.
5. M. Bhattacharyya, M. G. Schultz, E. Eskin, S. Hershkop, and S. J. Stolfo. MET: An Experimental System for Malicious Email Tracking. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, pages 1–12, September 2002.

6. J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.
7. J. Kannan, L. Subramanian, I. Stoica, and R. H. Katz. Analyzing Cooperative Containment of Fast Scanning Worms. In *Proceedings of Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, pages 17–23, July 2005.
8. H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the 13th USENIX Security Symposium*, pages 271–286, August 2004.
9. J. G. Levine, J. B. Grizzard, and H. L. Owen. Using Honeynets to Protect Large Enterprise Networks. *IEEE Security & Privacy*, 2(6):73–75, Nov/Dec 2004.
10. E. Levy. Approaching Zero. *IEEE Security & Privacy*, 2(4):65–66, July/August 2004.
11. M. Locasto, J. Parekh, S. Stolfo, A. Keromytis, T. Malkin, and V. Misra. Collaborative Distributed Intrusion Detection. Technical Report CUCS-012-04, Columbia University Department of Computer Science, 2004.
12. D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *Proceedings of 22nd Annual Joint Conference of IEEE Computer and Communication societies (INFOCOM)*, April 2003.
13. D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition*, April 2003.
14. P. Porras, L. Briesemeister, K. Levitt, J. Rowe, and Y.-C. A. Ting. A Hybrid Quarantine Defense. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 73–82, October 2004.
15. M. A. Rajab, F. Monrose, and A. Terzis. On the Effectiveness of Distributed Worm Monitoring. In *Proceedings of the 14th USENIX Security Symposium*, pages 225–237, August 2005.
16. S. Sidiroglou and A. D. Keromytis. A Network Worm Vaccine Architecture. In *Proceedings of the IEEE Workshop on Enterprise Technologies: Infrastructure for Collaborative Enterprises (WETICE), Workshop on Enterprise Security*, pages 220–225, June 2003.
17. S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proceedings of the 6th Symposium on Operating Systems Design & Implementation (OSDI)*, December 2004.
18. H. Toyozumi and A. Kara. Predators: Good Will Mobile Codes Combat against Computer Viruses. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, pages 13–21, September 2002.
19. J. Wu, S. Vangala, L. Gao, and K. Kwiat. An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, pages 143–156, February 2004.
20. V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *Proceedings of NDSS*, February 2004.
21. C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS)*, pages 190–199, October 2003.

An Authentication and Key Exchange Protocol for Secure Credential Services

SeongHan Shin¹, Kazukuni Kobara¹, and Hideki Imai^{1,2}

¹ Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST),
1-18-13, Sotokannda, Chiyoda-ku, Tokyo 101-0021 Japan
{seonghan.shin, k-kobara, h-imai}@aist.go.jp

² Chuo University, 1-13-27, Haruhi, Bunkyo-ku, Tokyo 112-8551 Japan

Abstract. In this paper, we propose a leakage-resilient and proactive authenticated key exchange (called LRP-AKE) protocol for credential services which provides not only a higher level of security against leakage of stored secrets but also secrecy of private key with respect to the involving server. The LRP-AKE protocol is provably secure in the random oracle model with the reduction to the computational Diffie-Hellman problem.

1 Introduction

The problem of safely storing client's private keys for a long period of time can be addressed with credential services. Consider a roaming client who accesses a network from different locations in order to retrieve his private keys associated with public keys for temporal use of PKI (for digital signature generation or public-key decryption). This kind of roaming protocol can be supported by a credentials server that authenticates the client and then assists in downloading private keys for the client.

The simplest roaming protocol is EAP-SIM [12] that specifies an EAP-based mechanism [8,2] using the GSM Subscriber Identity Module (SIM). The authenticity is based on secret keys stored on SIM and in an authentication server. EAP-SIM adds a physical security assumption for authentication, however some security flaws are discussed in [17]. Another roaming protocol is SPX LEAP [24], a tunneled EAP types such as TTLS (Tunneled TLS) and PEAP (Protected EAP), where a client transmits a password to a credential server through secure channels for authentication and then performs subsequent retrieval of the client's private key. This approach prevents off-line dictionary attacks at the sacrifice of using PKI. Ford and Kaliski [11] further described protocols that utilize multiple n servers, each of which holds a share of password-related data, in order to provide the protection of the credential server database. However, their protocols rely on a prior server-authenticated channel such as SSL which means it is vulnerable to web-spoofing attacks. Later, Jablon [13] proposed a password-only multi-server roaming protocol without need for prior secure channels.

Bellovin and Merritt [5] first introduced a secure password-only protocols where a client remembers a short password only (without any device and any additional assumption) and the corresponding server authenticates the client with the password or its verification data that is used to verify the client's knowledge of the password. By combining the roaming model and password-only protocols, Perlman and Kaufman [18] showed that simple modifications of the underlying password-only protocols are sufficient for secure roaming access to credentials. In order to integrate the convenience of password into the conventional PKI, two different approaches (called virtual soft token and virtual smartcard) have been proposed [18,14,19] in the name of password-enabled PKI. In the virtual soft token PKI [18,14], a private key encrypted with a password is stored on a server so that a client, after authenticating himself to the server and generating a strong session key, downloads the encrypted private key via the secure channel, decrypt it and use the private key as in the conventional PKI. In the virtual smartcard PKI [19], a client's private key is split into two parts (a password and a secret) where the latter is stored on a server. To perform a cryptographic operation (signature generation or decryption), the client first authenticates himself to the server using the password, generates a strong session key, and does the operation via the secure channel. On both approaches, Wang [26] proposed an intrusion-tolerant roaming protocol where a password verification data as well as a password-encrypted private key (or a partial secret) are shared among multiple servers using a threshold secret sharing scheme.

1.1 Motivation and Contribution

A more realistic threat on cryptographic techniques may be leakage of stored secrets that may be secret keys, private keys, password verification data and/or password-encrypted keys. When we consider a large number of password verification data and password-related credentials stored on a server, the leakage of either secret allows an adversary to mount off-line dictionary attacks enough to retrieve private keys. Such threat seems inevitable since, for example, an adversary might gain the `root` privilege of a server by exploiting bugs in server software. The other motivation is that the colluded servers in the password-only roaming protocols and password-enabled PKI can recover private keys with passwords that are easily deduced with off-line dictionary attacks. Note that, even if the functionality of one server is distributed to multiple servers, it doesn't help preventing collusion of all the servers. At the same time introducing multiple servers drastically increases the operational complexity. A particular concern is that the colluded servers generate thousands of digital signatures with large number of client's private keys (this indeed violates non-repudiation of digital signature!).

Keeping in mind the above motivation, we propose a leakage-resilient and proactive authenticated key exchange (LRP-AKE) protocol for credential services which provides not only a higher level of security against leakage of stored secrets

but also secrecy of private key with respect to the involving server.¹ The LRP-AKE protocol is provably secure in the random oracle model with the reduction to the computational Diffie-Hellman problem.

This paper is organized as follows. In Section 2, we propose a leakage-resilient and proactive authenticated key exchange (LRP-AKE) protocol. Section 3 and 4 are devoted to its model and security proof, followed by some discussions in Section 5. Finally, we conclude in Section 6.

2 A Leakage-Resilient and Proactive AKE Protocol

2.1 Preliminary

Let \mathbb{G} be a finite, cyclic group of prime order q and g be a generator of \mathbb{G} (quadratic residues modulo p where $p = aq+1$) where the Diffie-Hellman problem is hard. Let h be another generator of \mathbb{G} so that its discrete logarithm problem with g (i.e., computing $b = \log_g h$) should be hard. Both g and h may be given as public parameters. In the aftermath, all the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

Let k and l denote the security parameters for hash functions and public-key cryptosystems, respectively. Let N be a dictionary size of passwords. Let $\{0, 1\}^*$ denote the set of finite binary strings and $\{0, 1\}^l$ the set of binary strings of length l . Let "||" denote the concatenation of bit strings in $\{0, 1\}^*$. Let " \oplus " denote the exclusive-OR operation of bit strings. The hash functions are denoted $\mathcal{H}_j : \{0, 1\}^* \rightarrow \{0, 1\}^k$ for $j = 1, 2, 3, 4$ and 5 where \mathcal{H}_j are distinct secure one-way hash functions (e.g., SHA-256 or RIPEMD-160) one another. Let C and S be the identities of client and server, respectively, with each $ID \in \{0, 1\}^*$. Let $(K', Cert_C)$ be a credential where the former is a partial secret for client's private key K associated with a public key, which is included in the latter (i.e., the client's certificate) with other information.

2.2 The LRP-AKE Protocol

Consider a roaming client who has easy-to-be-lost/stolen devices with some memory capacity itself and wants to retrieve his private key with the help of a credential server equipped with its database, which might be insecure against possible attacks. In addition, neither PKI nor TRM is available.

The rationale of the LRP-AKE protocol² is that (i) client's password and additional secret are combined to be used for authentication; (ii) client's private

¹ The conceptual idea is similar to [22], based on RSA, in order to avoid reliance of PKI and TRM, and to deal with leakage of stored secrets in the model. In [25], Tang et al., showed an off-line dictionary attack on the protocol of [22]. The attack is possible due to the mistake of security model, and the corrected model and proof appeared in [23]. Of course, their attack is not possible in the LRP-AKE protocol.

² The main differences from [22] are its construction (based on the Diffie-Hellman protocol), behavior of Leak-query (only limited to C^i), security reduction (of course, to the CDH problem) and so on.

key is divided into two parts each of which is stored on client's or server's side; (iii) in order to provide proactive security of password and private key, the secrets are updated whenever client and server correctly run the protocol; (iv) a similar technique appeared in [3,1,10] is used in order to resist against off-line dictionary attacks after leakage of client's stored secrets. The LRP-AKE protocol consists of two phases: initialization and i -th ($i \geq 1$) protocol execution (see Fig. 1).

Initialization. In this phase, client C registers two kinds of secrets to server S in which one is used for authentication and generation of secure channels, and the other is for recovering the client's private key. First, the client chooses two random numbers s_1 from \mathbb{Z}_q^* and K_{11} from $\{0, 1\}^l$ where K_{11} is a partial secret and has the same size of his private key K . The client generates a verification data $W_1 \equiv h^{v_1}$, where $v_1 \equiv s_1 + pw \pmod q$, and the other partial secret K_{12} such that $K_{12} \leftarrow K_{11} \oplus K$. Then, client C registers securely W_1 and K_{12} along with his certificate $Cert_C$. At the end of this phase, the client stores secret value s_1 and one partial secret K_{11} on devices that may happen to leak the secrets, and remembers his password pw in mind. On the other hand, the server stores W_1 and $(K_{12}, Cert_C)$ on its database. Finally, they set a counter i as 1. The initialization is done only once.

i -th Protocol Execution. When client C wants to share a session key securely with server S in the i -th ($i \geq 1$) protocol execution, he should recover v_i by combining his password pw with secret value s_i stored on devices. The client chooses a random number x from \mathbb{Z}_q^* and computes the Diffie-Hellman public value $X \equiv g^x$. The latter is masked in a way of the product of the public value with the verification data h^{v_i} , and then sent to the server together with the client's identity and the counter. If i is not a correct counter, server S terminates the protocol. Otherwise, the server chooses a random number y from \mathbb{Z}_q^* and computes the Diffie-Hellman public value $Y \equiv g^y$. The server also computes X , by dividing the received masked public value with the verification data, and from the resultant value the Diffie-Hellman key $DHKey$ is derived. With the latter, server S easily generates an authenticator that is just the hash of some values and sends its Diffie-Hellman public value along with the server's identity and the authenticator. After computing $DHKey$ from Y , client C verifies the received authenticator V_S prior to generating his authenticator and a session key. Similarly, server S verifies the authenticator V_C before producing a secure channel with the session key. In order to avoid so-called partition attacks [16,28], both of client and server should check the subgroup order of Y and X^* , respectively: $Y^q \equiv (X^*)^q \equiv 1$.

Using the session key in a symmetric-key encryption (e.g., AES-CBC), server S sends the credential $(K_{i2}, Cert_C)$ to client C through the secure channel. Finally, the client recovers his private key from one partial secret (stored on own devices) and the other secret (received from the server) so that a pair of private key and certificate can be used as in usual PKI. If the verification of private key is valid, client C stores the next counter and newly-generated secrets that

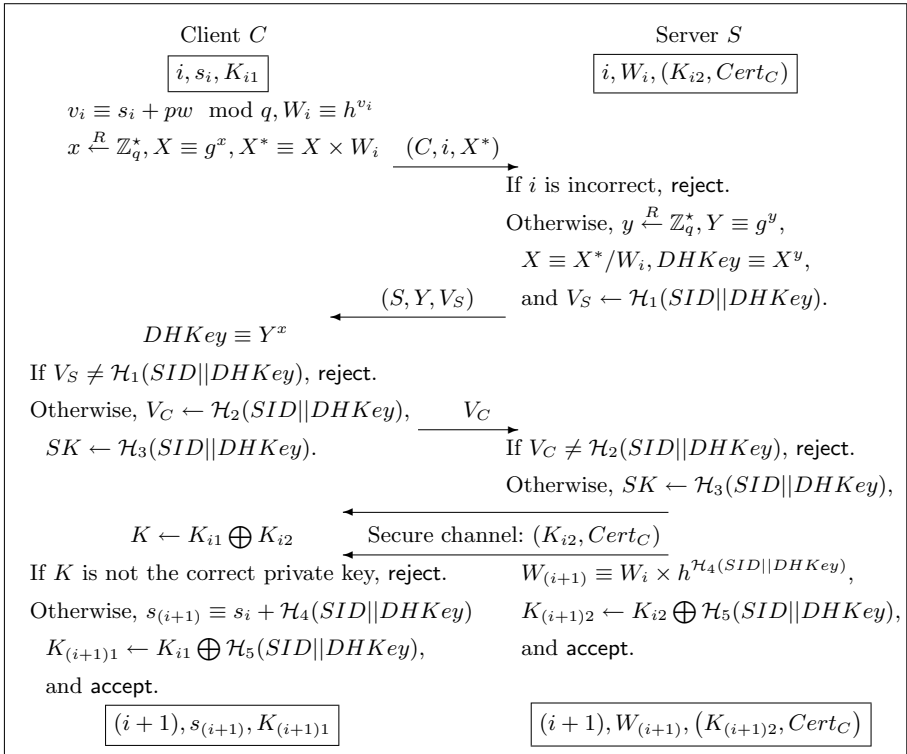


Fig. 1. The i -th ($i \geq 1$) protocol execution of the LRP-AKE protocol where $SID = C||S||i||X^*||Y||W_i$ plays a role of session identifier and the enclosed values in rectangle represent stored secrets of client and server, respectively

are refreshed from the hash of the Diffie-Hellman key and other information. In the same way, server S stores the next counter and refreshed secrets on its database. These stored secrets will be used for the next session between the parties without changing the client’s password. Note that the frequent change of passwords might incur the risk of password to be exposed, simply because people tends to write it down on somewhere or needs considerable efforts to remember new passwords. The hash functions \mathcal{H}_4 and \mathcal{H}_5 used here can be replaced by pseudo random functions to produce appropriate output sizes.

3 The Model and Security Notions

Here we introduce an extended model based on [6,4] and security notions.

The Model. We denote by C and S two parties that participate in the key exchange protocol P . Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of P where we denote C

(resp., S) instances by C^i (resp., S^j), or by U in case of any instance. During the execution of P , an adversary has the entire control of the network and additionally has access to the parties' stored secrets where the latter simulates *insecure* devices and databases. Let us show the capability of adversary \mathcal{A} each query captures:

- **Execute**(C^i, S^j): This query models passive attacks, where the adversary gets access to honest executions of P between C^i and S^j by eavesdropping.
- **Send**(U, m): This query models active attacks by having \mathcal{A} send a message to instance U . The adversary \mathcal{A} gets back the response U generates in processing the message m according to the protocol P . A query **Send**(C^i, Start) initializes the key exchange protocol.
- **Reveal**(U): This query handles the misuse of the session key by any instance U . The query is only available to \mathcal{A} if the instance actually holds a session key and the latter is released to \mathcal{A} .
- **Leak**(U): This query handles the leakage of the "stored" secrets by any instance U . The adversary \mathcal{A} gets back (s_i, K_{i1}) and $(W_i, (K_{i2}, \text{Cert}_C))$ where the former (resp., the latter) is released if the instance corresponds to C^i (resp., S^j).
- **Test**(U): The **Test**-query can be asked at most once by the adversary \mathcal{A} and is only available to \mathcal{A} if the instance U is "fresh" in that the session key is not obviously known to the adversary. This query is answered as follows: one flips a (private) coin $b \in \{0, 1\}$ and forwards the corresponding session key SK (**Reveal**(U) would output) if $b = 1$, or a random value except the session key if $b = 0$.

Security Notions. The adversary \mathcal{A} is provided with random coin tosses, some oracles and then is allowed to invoke any number of queries as described above, in any order. The aim of the adversary is to break the privacy of the session key or the authentication of the parties in the context of executing P . The AKE security is defined by the game $\mathbf{Game}^{\text{ake}}(\mathcal{A}, P)$, in which the ultimate goal of the adversary is to guess the bit b involved in the **Test**-query by outputting this guess b' . We denote the AKE advantage, by $\text{Adv}_P^{\text{ake}}(\mathcal{A}) = 2 \Pr[b = b'] - 1$, as the probability that \mathcal{A} can correctly guess the value of b . The protocol P is said to be (t, ε) -AKE-secure if \mathcal{A} 's advantage is smaller than ε for any adversary \mathcal{A} running time t .

Another goal is to consider unilateral authentication of either C (C -auth) or S (S -auth) wherein the adversary impersonates a party. We denote by $\text{Succ}_P^{C\text{-auth}}(\mathcal{A})$ (resp., $\text{Succ}_P^{S\text{-auth}}(\mathcal{A})$) the probability that \mathcal{A} successfully impersonates an C instance (resp., an S instance) in an execution of P , which means that S (resp., C) agrees on a key while the latter is shared with no instance of C (resp., S). A protocol P is said to be (t, ε) -Auth-secure if \mathcal{A} 's success for breaking either C -auth or S -auth is smaller than ε for any adversary \mathcal{A} running time t .

3.1 Computational Diffie-Hellman Assumption

A (t, ε) -CDH $_{g, \mathbb{G}}$ attacker, in a finite cyclic group \mathbb{G} of prime order q with g as a generator, is a probabilistic machine \mathcal{B} running in time t such that its success

probability $\text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(\mathcal{B})$, given random elements g^x and g^y to output g^{xy} , is greater than ε . We denote by $\text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t)$ the maximal success probability over every adversaries running within time t . The CDH-Assumption states that $\text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t) \leq \varepsilon$ for any t/ε not too large.

4 Security

In this section we show that the LRP-AKE protocol of Fig. 1. is provably secure in the random oracle model³ [7].

In order to simplify the security proof, we only consider the first two flows of the i -th protocol execution (unilateral authentication of S to C) as in [6,4]. In the proof, we restrict the Leak-query to instance C^i only. This is due to the fact that in a 2-party password-based AKE protocol one cannot prove the security when an adversary gets the verification data. However, the Leak-query to instance S^j would be considered for discussing about the security of the client's password and private key. For the sake of brevity, we also omit the index i in the proof.

Theorem 1. (AKE/UA Security) *Let P be the LRP-AKE protocol of Fig. 1., where passwords are chosen from a dictionary of size N . For any adversary \mathcal{A} within a polynomial time t , with less than q_s active interactions with the parties (Send-queries), q_p passive eavesdroppings (Execute-queries) and asking q_h hash queries to any \mathcal{H}_j , $\text{Adv}_P^{\text{ake}}(\mathcal{A}) \leq 4\varepsilon$ and $\text{Adv}_P^{S\text{-auth}}(\mathcal{A}) \leq \varepsilon$, with ε upper-bounded by*

$$\frac{3q_s}{N} + 3q_h^2 \times \text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t + 3\tau_e) + \frac{q_s}{2^{k_1}} + \frac{8q_s + 2q_p + (q_s + q_p)^2}{2q}, \tag{1}$$

where k_1 is the output length of \mathcal{H}_1 and τ_e denotes the computational time for an exponentiation in \mathbb{G} .

The proof appears in Appendix A. This theorem shows that the LRP-AKE protocol is secure against dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions (number of Send-queries) to the number of passwords. Note that in order to do on-line dictionary attacks successively the adversary should obtain the client's devices whenever refreshment of stored secrets happens. This is of practical significance since in the real world applications the leakage of stored secrets is limited by the *physical* power of the adversary which are usually much less.

Now we consider about security of password and private key against the leakage of stored secrets from client and server, respectively.

³ Security in the random oracle model is only a heuristic: it does not imply security in the real world [9]. Nevertheless, the random oracle model is a useful tool for validating natural cryptographic constructions. Security proofs in this model prove security against adversaries that are confined to the random oracle world.

Theorem 2. *The password and the private key in the LRP-AKE protocol is information-theoretically secure with respect to the leakage of stored secrets from client and server, respectively.*

If an adversary gets the client's stored secrets (s_i and K_{i1}), the secrets don't reveal any information about the password and the private key simply because s_i is completely independent from the password and K_{i1} is one share of (2, 2)-threshold perfect secret sharing scheme. In the case where an adversary gets the server's stored secrets (W_i and $(K_{i2}, Cert_C)$), the adversary can freely impersonate server \mathcal{S} with W_i . We cannot avoid this impersonation attack as all of the 2-party authentication protocols. Nevertheless, W_i and K_{i2} don't reveal any information about the password and the private key with the same reason as above.

Theorem 3. *The LRP-AKE protocol provides proactive security of password and private key.*

In the context of executing the LRP-AKE protocol, it is obvious. One may regard this protocol as an instantiation of [15]. Suppose that the client and the server run the LRP-AKE protocol at a fixed period of time (e.g., a day) and an adversary obtains the secrets (W_i and K_{i2}) at that time. If the parties complete refreshing the secrets (e.g., in the next day) before the adversary does, the latter of course cannot do the impersonation attack any more.

5 Discussions

At first, one can notice that the LRP-AKE protocol doesn't allow even on-line dictionary attacks without any leakage of stored secrets since the authentication depends on the strong secret s_i as in [7,20].

A more interesting application of the LRP-AKE protocol may be a Grid computing environment where a client obtains different short-term credentials (e.g., short-term X.509 certificates or proxy certificates), which can be subsequently used for the access of other services on the Grid [27], from servers after successful authentication with only one password. In the previous password-based roaming protocols (e.g., SPX LEAP, [18,14,19]), a compromise of one server unfortunately results in the total insecurity of the remaining servers since the adversary can get the password (i.e., the only secret for authentication) through off-line dictionary attacks on the password verification data. On the other hand, this application can be realized by the LRP-AKE protocol due to the fact that both the password and the partial credential are completely protected from the leakage of server's stored secrets. This also implies that both the verification data and many partial credentials don't need to be distributed or shared among multiple servers.

Compared to the password-only roaming protocols and password-enabled PKI, the stateful storage of client in the LRP-AKE protocol may be a strong assumption

but it is a weak assumption rather than requiring TRM. As we pointed out in Section 1.1, the previous protocols have vulnerability against the leakage of stored secrets on client side (e.g., EAP-SIM and smartcards) and don't guarantee secrecy of private key against an insider adversary even if the client's private key is encrypted with password or distributed among multiple servers. Remind that, without PKI and TRM, the LRP-AKE protocol not only has resilience against the leakage of client's stored secrets but also provides secrecy of private key against the leakage of server's stored secrets. Actually, the LRP-AKE protocol has a sort of two-factor protection for the private key: client needs a password and additional secret to download the partial credential, and the other partial key stored on client's devices is used to retrieve the (full) private key. Though holding insecure devices may entail inconvenience to clients, we can stress that it is significantly applicable to the real world (e.g., think of a client who carries mobile devices, such as mobile phones or PDAs, with some memory capacity) and it is a small price to pay for a higher level of security against leakage of stored secrets (including several security properties) and more efficiency over the previous ones [11,13,26] in terms of computation costs and communication bandwidth among multiple servers.

6 Conclusions

In this paper, we have proposed a leakage-resilient and proactive authenticated key exchange (LRP-AKE) protocol for credential services which not only improves significantly security against leakage of stored secrets but also provides secrecy of private key with respect to the involving server. We also proved its security of the LRP-AKE protocol in the random oracle model with the reduction to the computational Diffie-Hellman problem. The LRP-AKE protocol is conceptually similar to [22], but remember that an RSA-based AKE protocol cannot be necessarily translated to a secure Diffie-Hellman based one.

Acknowledgements

The authors appreciate anonymous reviewers for their helpful comments. This research has been sponsored by the Ministry of Economy, Trade and Industry (METI), Japan under the contract of "New Generation of Information Security R&D Program".

References

1. M. Abdalla, E. Bresson, O. Chevassut, B. Moller, and D. Pointcheval. Provably Secure Password-based Authentication in TLS. In *Proc. of AsiaCCS 2006*, ACM, 2006.
2. B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson and H. Levkowetx. Extensible Authentication Protocol (EAP). IETF RFC 3748, June 2004.
3. M. Abdalla and D. Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. In *Proc. of CT-RSA 2005*, LNCS 3376, pages 191-208. Springer-Verlag, 2005.

4. E. Bresson, O. Chevassut, and D. Pointcheval. New Security Results on Encrypted Key Exchange. In *Proc. of PKC 2004*, LNCS 2947, pages 145-158. Springer-Verlag, 2004.
5. S. M. Bellare and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of IEEE Symposium on Security and Privacy*, pages 72-84, 1992.
6. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Proc. of EUROCRYPT 2000*, LNCS 1807, pages 139-155. Springer-Verlag, 2000.
7. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS '93*, pages 62-73, 1993.
8. L. Blunk and J. Vollbrecht. PPP Extensible Authentication Protocol (EAP). IETF RFC 2284, March 1998.
9. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *Proc. of the 30th ACM Symposium on Theory of Computing (STOC)*, pages 209-218, ACM, 1998.
10. D. Catalano, D. Pointcheval, and T. Pornin. Trapdoor Hard-to-Invert Group Isomorphisms and Their Application to Password-based Authentication. *Journal of Cryptology*, 2006. The extended abstract appeared at CRYPTO 2004.
11. W. Ford and B. S. Kaliski. Server-Assisted Generation of a Strong Secret from a Password. In *Proc. of the Fifth International Workshop on Enterprise Security*, IEEE, 2000.
12. H. Haverinen and J. Salowey. Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM), December 2004.
13. D. Jablon. Password Authentication Using Multiple Servers. In *Proc. of CT-RSA 2001*, LNCS 2020, pages 344-360. Springer-Verlag, 2001.
14. T. Kwon. Virtual Software Tokens - A Practical Way to Secure PKI Roaming. In *Proc. of the Infrastructure Security (InfraSec)*, LNCS 2437, pages 288-302. Springer-Verlag, 2002.
15. R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In *Proc. of 10th Annual ACM Symposium on Principles of Distributed Computing*, 1991.
16. S. Patel. Number Theoretic Attacks on Secure Password Schemes. In *Proc. of IEEE Symposium on Security and Privacy*, IEEE Computer Society, pages 236-247, 1997.
17. S. Patel. Analysis of EAP-SIM Session Key Agreement. available at <http://www.drizzle.com/~aboba/EAP/AnalysisOfEAP.pdf>
18. R. Perlman and C. Kaufman. Secure Password-Based Protocol for Downloading a Private Key. In *Proc. 1999 Network and Distributed System Security Symposium*, Internet Security, 1999.
19. R. Sandhu, M. Bellare, and R. Ganesan. Password Enabled PKI: Virtual Smartcards vs. Virtual Soft Tokens. In *Proc. of the 1st Annual PKI Research Workshop*, pages 89-96, 2002.
20. V. Shoup. On Formal Models for Secure Key Exchange. IBM Research Report RZ 3121, 1999. Available at <http://eprint.iacr.org/1999/012>.
21. V. Shoup. OAEP Reconsidered. *Journal of Cryptology*, Vol. 15(4), pages 223-249, September 2002.
22. S. H. Shin, K. Kobara, and H. Imai. Efficient and Leakage-Resilient Authenticated Key Transport Protocol Based on RSA. In *Proc. of ACNS 2005*, LNCS 3531, pages 269-284. Springer-Verlag, 2005.
23. S. H. Shin, K. Kobara, and H. Imai. Security Proof of "Efficient and Leakage-Resilient Authenticated Key Transport Protocol Based on RSA". *Cryptology ePrint Archive*, Report 2005/190, 2005.

24. J. Tardo and K. Alagappan. SPX: Global Authentication Using Public Key Certificates. In *Proc. of 1991 IEEE Computer Society Symposium on Security and Privacy*, pages 232-244, 1991.
25. Q. Tang and C. J. Mitchell. Weaknesses in a Leakage-Resilient Authenticated Key Transport Protocol. Cryptology ePrint Archive, Report 2005/173, 2005.
26. X. Wang. Intrusion-Tolerant Password-Enabled PKI. In *Proc. of the 2nd Annual PKI Research Workshop*, pages 44-53, 2003.
27. V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder and F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *Proc. of the 3rd Annual PKI R&D Workshop*, 2004.
28. Z. Wan and S. Wang. Cryptanalysis of Two Password-Authenticated Key Exchange Protocols. In *Proc. of ACISP 2004*, LNCS 3108, pages 164-175. Springer-Verlag, 2004.

A Proof of Theorem 1

Proof. In this proof, we incrementally define a sequence of games starting at the real game \mathbf{G}_0 and ending up at \mathbf{G}_5 . We use Shoup’s lemma [21] to bound the probability of each event in these games.

Game \mathbf{G}_0 (Real protocol): This is the real protocol in the random oracle model. We are interested in the following two events:

- S_0 (for semantic security) which occurs if the adversary correctly guesses the bit b involved in the **Test**-query;
- A_0 (for S -authentication) which occurs if an instance C^i accepts with no partner instance S^j with the same transcript $((C, X^*), (S, Y, V_S))$

$$\text{Adv}_P^{\text{ake}}(\mathcal{A}) = 2 \Pr[S_0] - 1 \quad \text{Adv}_P^{S\text{-auth}}(\mathcal{A}) = \Pr[A_0] . \quad (2)$$

In any game \mathbf{G}_n below, we study the event A_n and the restricted event $\text{Sw}A_n = S_n \wedge \neg A_n$.

Game \mathbf{G}_1 (Simulation of hash and other queries): In this game, we simulate the hash oracles (\mathcal{H}_1 and \mathcal{H}_3 , but as well additional hash functions, for $j = 1, 3, \mathcal{H}'_j : \{0, 1\}^* \rightarrow \{0, 1\}^{k_j}$ that will appear in the Game \mathbf{G}_3) as usual by maintaining hash lists $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{H}'}$ (see below). We also simulate all the instances, as the real parties would do, for the **Leak**-queries and for the **Send**, **Execute**, **Reveal** and **Test**-queries (see below). From this simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

Simulation of the hash functions: \mathcal{H}_j oracles

- For a hash-query $\mathcal{H}_j(q)$ (resp., $\mathcal{H}'_j(q)$), such that a record (j, q, r) appears in $\Lambda_{\mathcal{H}}$ (resp., $\Lambda_{\mathcal{H}'}$), the answer is r . Otherwise one chooses a random element $r \xleftarrow{R} \{0, 1\}^{k_j}$, answers with it, and adds the record (j, q, r) to $\Lambda_{\mathcal{H}}$ (resp., $\Lambda_{\mathcal{H}'}$).

Simulation of the LRP-AKE protocol

Send-queries to C

We answer to the **Send**-queries to a C -instance as follows:

- A **Send**(C^i, Start)-query is processed according to the following rules:
 - ▶ **Rule C1**⁽¹⁾
Choose random elements $(b, \nu, \omega) \xleftarrow{R} (\mathbb{Z}_q^*)^3$, and compute $h \equiv g^b$ and $W \equiv h^\nu$ such that $s \equiv \nu - \omega \pmod q$.
 - ▶ **Rule C2**⁽¹⁾
Choose a random element $\theta \xleftarrow{R} \mathbb{Z}_q^*$, and compute $X \equiv g^\theta$ and $X^* \equiv X \times W$.

Then the query is answered with (C, X^*) , and the instance goes to an expecting state.

- If the instance C^i is in an expecting state, a query **Send**($C^i, (S, Y, V_S)$) is processed by computing the alleged authenticator and the session key. We apply the following rules.

- ▶ **Rule C3**⁽¹⁾
Compute $DHKey \equiv Y^\theta$.
- ▶ **Rule C4**⁽¹⁾

Compute the expected authenticator and the session key:

$$V'_S \leftarrow \mathcal{H}_1(C||S||X^*||Y||W||DHKey),$$

$$SK_C \leftarrow \mathcal{H}_3(C||S||X^*||Y||W||DHKey).$$

If $V'_S = V_S$, then the instance accepts. In any case, it terminates.

Send-queries to S

We answer to the **Send**-queries to a S -instance as follows:

- A **Send**($S^j, (C, X^*)$)-query is processed according to the following rule:
 - ▶ **Rule S1**⁽¹⁾
Choose a random element $\varphi \xleftarrow{R} \mathbb{Z}_q^*$ and compute $Y \equiv g^\varphi$.

Then, the instance computes the authenticator and the session key. We apply the following rules:

- ▶ **Rule S2**⁽¹⁾
Compute $X \equiv X^*/W$ and $DHKey \equiv X^\varphi$.
- ▶ **Rule S3**⁽¹⁾

Compute the authenticator and the session key:

$$V_S \leftarrow \mathcal{H}_1(C||S||X^*||Y||W||DHKey),$$

$$SK_S \leftarrow \mathcal{H}_3(C||S||X^*||Y||W||DHKey).$$

Finally, the query is answered with (S, Y, V_S) , and then the instance accepts and terminates.

Other queries

- An **Execute**(C^i, S^j)-query is processed using successively the above simulations of the **Send**-queries: $(C, X^*) \leftarrow \text{Send}(C^i, \text{Start})$, $(S, Y, V_S) \leftarrow \text{Send}(S^j, (C, X^*))$, and then outputting the transcript $((C, X^*), (S, Y, V_S))$.
- A **Reveal**(U)-query returns the session key (SK_C or SK_S) computed by the instance U (if the former has accepted).
- A **Leak**(C^i)-query returns the secret s computed by the instance C .
- A **Test**(U)-query first gets SK from **Reveal**(U), and flip a coin b . If $b = 1$, we return the value of the session key SK , otherwise we return a random value drawn from $\{0, 1\}^{k_3}$.

Game \mathbf{G}_2 (Collisions): For an easier analysis in the following, we cancel games in which some collisions (Coll_2) are unlikely to happen:

- collisions on the partial transcripts $((C, X^*), (S, Y))$: any adversary tries to find out one pair (X^*, Y) , coinciding with the challenge transcript, and then obtain the corresponding session key using the *Reveal*-query. However, at least one party involves with the transcripts, and thus one of X^* and Y is truly uniformly distributed.

The probability is bounded by the birthday paradox:

$$\Pr[\text{Coll}_2] \leq \frac{(q_p + q_s)^2}{2q} . \tag{3}$$

Game \mathbf{G}_3 (Using private oracles): In order to make the authenticator and the session key unpredictable to any adversary, we compute them using the private oracles \mathcal{H}'_1 and \mathcal{H}'_3 (instead of \mathcal{H}_1 and \mathcal{H}_3), respectively, so that the values are completely independent from the random oracles. We reach this aim by using the following rule:

► **Rule C4/S3**⁽³⁾

Compute the authenticator $V_S \leftarrow \mathcal{H}'_1(C||S||X^*||Y)$.

Compute the session key $SK_{C/S} \leftarrow \mathcal{H}'_3(C||S||X^*||Y)$.

Since we do no longer need to compute the values *DHKey*, we can simplify the following rules:

► **Rule C3/S2**⁽³⁾

Do nothing.

Finally, the secret W is not used anymore either so that we can also simplify the generation of X^* using the group property of \mathbb{G} .

► **Rule C2**⁽³⁾

Choose a random element $x \xleftarrow{R} \mathbb{Z}_q^*$ and compute $X^* \equiv g^x$.

The games \mathbf{G}_3 and \mathbf{G}_2 are indistinguishable unless some specific hash queries are asked, denoted by event $\text{AskH}_3 = \text{AskH3w1}_3 \vee \text{AskH1}_3$:

- AskH1_3 : $\mathcal{H}_1(C||S||X^*||Y||W||DHKey)$ has been queried by \mathcal{A} to \mathcal{H}_1 for some execution transcripts $((C, X^*), (S, Y, V_S))$;
- AskH3w1_3 : $\mathcal{H}_3(C||S||X^*||Y||W||DHKey)$ has been queried by \mathcal{A} to \mathcal{H}_3 for some execution transcripts $((C, X^*), (S, Y, V_S))$, where some party has accepted, but event AskH1_3 did not happen.

The authenticator is computed with a random oracle that is private to the simulator, then one can remark that it cannot be guessed by the adversary, better than at random for each attempt, unless the same partial transcript $((C, X^*), (S, Y))$ appeared in another session with a real instance S^j . But such a case has already been excluded (in Game \mathbf{G}_2). A similar remark holds on the session key:

$$\Pr[A_3] \leq \frac{q_s}{2^{k_1}} \quad \Pr[\text{SwA}_3] = \frac{1}{2} . \tag{4}$$

When collisions of the partial transcripts have been excluded, the event AskH1 can be split in three disjoint sub-cases:

- AskH1-Passive₃: the transcript $((C, X^*), (S, Y, V_S))$ comes from an execution between instances of C and S (Execute-queries or forward of Send-queries, relay of part of them). This means that both X^* and Y have been simulated;
- AskH1-WithC₃: the execution involved an instance of C , but Y has not been sent by any instance of S . This means that X^* has been simulated, but Y has been produced by the adversary;
- AskH1-WithS₃: the execution involved an instance of S , but X^* has not been sent by any instance of C . This means that Y has been simulated, but X^* has been produced by the adversary.

Game G₄ (Introduction of DH instance): In order to evaluate the above events, we introduce a random Diffie-Hellman instance (P, Q) (where both P and Q are generators of \mathbb{G} . Otherwise, the Diffie-Hellman problem is easy.) We first modify the simulation of the party C , involving the element Q .

► **Rule C1**⁽⁴⁾

Choose random elements $(v, w) \xleftarrow{R} (\mathbb{Z}_q^*)^2$ and compute $W \equiv Q^v$ such that $s \equiv v - w \pmod q$.

By the isomorphic property from \mathbb{G}^* to \mathbb{G}^* , the new W is perfectly indistinguishable from before since there exists a unique discrete logarithm for W . We also introduce the other part P of the Diffie-Hellman instance in the simulation of the party S .

► **Rule S1**⁽⁴⁾

Choose a random element $y \xleftarrow{R} \mathbb{Z}_q^*$ and compute $Y \equiv P^y$.

It would let the probabilities unchanged, but note that we excluded the cases $W \equiv 1$ and $Y \equiv 1$:

$$|\Pr[\text{AskH}_4] - \Pr[\text{AskH}_3]| \leq \frac{q_s + q_p}{q}. \tag{5}$$

Game G₅ (Collision of W and Probability of AskH): It is now possible to evaluate the probability of the event AskH (or more precisely, the sub-cases). Indeed, one can see that the password is never used during the simulation. It does not need to be chosen in advance, but at the very end only. Then, an information-theoretic analysis can be done which simply uses cardinalities of some sets.

To this aim, we first cancel a few more games, wherein for some pairs $(X^*, Y) \in \mathbb{G}^2$, involved in a communication between an instance S^j and either the adversary or an instance C^i , there are two distinct elements W such that the tuple $(X^*, Y, W, \text{CDH}_{g, \mathbb{G}}(X^*/W, Y))$ is in $\Lambda_{\mathcal{H}}$ (which event is denoted CollH₅):

$$|\Pr[\text{AskH}_5] - \Pr[\text{AskH}_4]| \leq \Pr[\text{CollH}_5]. \tag{6}$$

The event CollH₅ can be upper-bounded, granted the following lemma:

Lemma 1. *If for any pair $(X^*, Y) \in \mathbb{G}^2$, involved in a communication with an instance S^j , there are two elements W_0 and W_1 such that $(X^*, Y, W_i, Z_i =$*

$\text{CDH}_{g,\mathbb{G}}(X^*/W_i, Y)$) is in $\Lambda_{\mathcal{H}}$, one can solve the computational Diffie-Hellman problem:

$$\Pr[\text{CollH}_5] \leq q_h^2 \times \text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t + \tau_e) . \tag{7}$$

Proof. We assume that there exist $(X^*, Y \equiv P^y) \in \mathbb{G}^2$ involved in a communication with an instance S^j , and two elements $W_0 \equiv Q^{v_0}$ and $W_1 \equiv Q^{v_1}$ such that the tuple $(X^*, Y, W_i, Z_i = \text{CDH}_{g,\mathbb{G}}(X^*/W_i, Y))$ is in $\Lambda_{\mathcal{H}}$, for $i = 0, 1$. Then,

$$Z_i = \text{CDH}_{g,\mathbb{G}}(X^*/W_i, Y) = \text{CDH}_{g,\mathbb{G}}(X^*, Y) \times \text{CDH}_{g,\mathbb{G}}(P, Q)^{y(-v_i)} . \tag{8}$$

As a consequence, $Z_1/Z_0 = \text{CDH}_{g,\mathbb{G}}(P, Q)^{y(v_0-v_1)}$, and thus $\text{CDH}_{g,\mathbb{G}}(P, Q) = (Z_1/Z_0)^\psi$ where ψ is the inverse of $y(v_0 - v_1)$ in \mathbb{Z}_q^* . The latter exists since $W_0 \neq W_1$ and $y \neq 0$. By guessing the two queries asked to the \mathcal{H}_j , one concludes the proof. \square

In order to conclude the proof, let us study separately the three sub-cases of AskH1 and then AskH3w1 (keeping in mind the absence of several kinds of collisions: for partial transcripts, and for W in \mathcal{H} -queries):

- AskH1-Passive: About the passive transcripts (in which both X^* and Y have been simulated), one can state the following lemma:

Lemma 2. *If for any pair $(X^*, Y) \in \mathbb{G}^2$, involved in a passive transcript, there is an element W such that $(X^*, Y, W, Z = \text{CDH}_{g,\mathbb{G}}(X^*/W, Y))$ is in $\Lambda_{\mathcal{H}}$, one can solve the computational Diffie-Hellman problem:*

$$\Pr[\text{AskH1-Passive}_5] \leq q_h \times \text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t + 2\tau_e) . \tag{9}$$

Proof. We assume that there exist $(X^* \equiv g^x, Y \equiv P^y) \in \mathbb{G}^2$ involved in a passive transcript and $W \equiv Q^v$ such that the tuple (X^*, Y, W, Z) is in $\Lambda_{\mathcal{H}}$. As above,

$$Z = \text{CDH}_{g,\mathbb{G}}(X^*, Y) \times \text{CDH}_{g,\mathbb{G}}(Q, Y)^{-v} = P^{xy} \times \text{CDH}_{g,\mathbb{G}}(P, Q)^{-yv} . \tag{10}$$

As a consequence, $\text{CDH}_{g,\mathbb{G}}(P, Q) = (Z/P^{xy})^\psi$ where ψ is the inverse of $-yv$ in \mathbb{Z}_q^* . The latter exists since we have excluded the cases where $y = 0$ or $v = 0$. By guessing the query asked to the \mathcal{H}_j , one concludes the proof. \square

- AskH1-WithC: This corresponds to an attack where the adversary tries to impersonate S to C (break unilateral authentication). But each authenticator sent by the adversary (who may have obtained the secret s in the Leak-query, denoted by event Leak₅) has been computed with at most one W value:

$$\Pr[\text{AskH1-WichC}_5] \leq \frac{q_s}{q} + \frac{q_s}{N} . \tag{11}$$

- AskH1-WithS: The above Lemma 1, applied to games where the event CollH₅ did not happen, states that for each pair (X^*, Y) involved in a transcript with an instance S^j , there is at most one element W such that for $W \equiv Q^s \times Q^{pw}$,

the corresponding tuple is in $\mathcal{A}_{\mathcal{T}}$: the probability for the adversary (who may have obtained the secret s in the Leak-query) over a random password is as above:

$$\Pr[\text{AskH1-WithS}_5] \leq \frac{q_s}{q} + \frac{q_s}{N} . \tag{12}$$

About AskH3w1 (when the above three events did not happen), it means that only executions with an instance of S (and either C or the adversary) may lead to acceptance. Exactly the same analysis as for AskH1-Passive and AskH1-WithS leads to

$$\Pr[\text{AskH3w1}_5] \leq \frac{q_s}{q} + \frac{q_s}{N} + q_h \times \text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t + 2\tau_e) . \tag{13}$$

As a conclusion, we get an upper-bound for the probability of AskH₅ by combining all the cases:

$$\Pr[\text{AskH}_5] \leq \frac{3q_s}{q} + \frac{3q_s}{N} + 2q_h \times \text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t + 2\tau_e) . \tag{14}$$

Combining equation (3), (4), (5), (7) and (14), one gets either

$$\Pr[\mathbf{A}_0] \leq \frac{q_s}{2^{k_1}} + \Delta \quad \Pr[\text{SwA}_0] = \frac{1}{2} + \Delta , \tag{15}$$

where

$$\Delta \leq \frac{3q_s}{N} + 3q_h^2 \times \text{Succ}_{g,\mathbb{G}}^{\text{cdh}}(t + 3\tau_e) + \frac{8q_s + 2q_p + (q_s + q_p)^2}{2q} . \tag{16}$$

One can get the result as desired by noting that $\Pr[\mathbf{S}_0] \leq \Pr[\text{SwA}_0] + \Pr[\mathbf{A}_0]$.

A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders

Yvo Desmedt^{1,*}, Josef Pieprzyk², Ron Steinfeld², and Huaxiong Wang²

¹ Department of Computer Science,
University College London, UK

² Centre for Advanced Computing – Algorithm and Cryptography,
Department of Computing,
Macquarie University, Australia

Abstract. In this paper we make progress towards solving an open problem posed by Katz and Yung at CRYPTO 2003. We propose the first protocol for key exchange among $n \geq 2k + 1$ parties which simultaneously achieves all of the following properties:

1. Key Privacy (including forward security) against active attacks by group *outsiders*,
2. Non-malleability — meaning in particular that no subset of up to k corrupted group *insiders* can ‘fix’ the agreed key to a desired value, and
3. Robustness against denial of service attacks by up to k corrupted group *insiders*.

Our insider security properties above are achieved assuming the availability of a reliable broadcast channel.

1 Introduction

An Authenticated Key Exchange (AKE) protocol is a fundamental public-key cryptographic tool which allows a group of parties to establish a common secret *session key* over an insecure network. This session key can then be used to allow confidential and authenticated communication among the group members using symmetric cryptography.

The most important security requirement for an AKE protocol is *privacy against group outsiders*. Informally, this requirement prevents any outsider party not belonging to the group of parties participating in the protocol from getting any information on the session key, even if the outsider can observe and modify protocol messages sent by group parties, or inject new messages into the network.

Recently Katz and Yung [13] presented the first constant-round AKE protocol provably secure against active group outsiders in the standard model under known cryptographic assumptions. However, they left open the problem of constructing an AKE protocol which also offers security against attack by malicious *group insiders*. In this paper, we make progress towards a solution to this problem. We consider two types of insider attacks:

* The author is BT Professor of Information Security and a courtesy professor at the Department of Computer Science, Florida State University, USA.

1. *Denial of Service Attacks*, in which the malicious group insiders send ‘bad’ protocol messages in order to abort an honest party or cause two honest parties to compute different session keys.
2. *Malleability Attacks*, in which the malicious group insiders send ‘bad’ protocol messages in order to bias the probability distribution of the session key or even force the session key to some desired value. Such an attack by the insiders can allow a collaborating outsider to know some information on the session key even if the insiders are *shielded*, i.e. prevented from communicating with the outsider after the protocol session begins.

To deal with an inside adversary who tries to use *covert channels* we need to define formal *robustness* and ‘*shielded-insider*’ *privacy* notions to capture security of an AKE protocol against the denial of service and malleability attacks, respectively. Both these notions assume a setting in which up to k out of $n \geq 2k + 1$ insiders are corrupted, and all players have access to a reliable public broadcast channel. We then present a constant-round AKE protocol and prove that it achieves these *insider security* notions in the standard model, under known assumptions, as well as achieving privacy against group *outsiders*.

This paper is organized as follows. Section 1.1 compares our work with previous results. In Section 2 we summarize notation and building blocks we use. We define the security models for both insider and outsider security in Section 3. In Section 4 we present a basic unauthenticated version of our protocol constructed from any public-key encryption scheme and prime-order group in which the discrete-log problem is hard. Section 5 contains a statement of our security results for the basic protocol. In Section 6 we discuss how we strengthen the basic protocol against active attacks by adding authentication. Due to page limits, we have omitted the security proofs from this version of the paper. They are included in the full version of the paper, available from the authors.

1.1 Related Work

Previous work on group AKE protocols has focussed on outsider security, whereas our protocol achieves both outsider and insider security goals in a constant number of rounds. Thus our work can be viewed as an extension of the work of Katz and Yung [13] and Boyd and Nieto [3], who both described constant-round protocols secure against outsider attacks. Earlier work on AKE protocols includes [6,4,5]. All these previous protocols, however, are not designed to be secure against ‘malleability attack’ by malicious insiders, who can choose their protocol messages, based on the observed messages of honest parties, in order to bias the session key (see section 3 for further details).

In an independent work from ours, Katz and Sun Shin [12] define insider security for group key exchange protocols and show a generic compiler to transform any AKE protocol to an insider-secure one. Our model of insider security differs from the one in [12] both in the assumptions made and in the security achieved (i.e. the types of attacks which are prevented). With regard to assumptions, the main difference is that our insider security model assumes a reliable broadcast channel, whereas [12] does not make this assumption, allowing the

attacker to modify or block any communications. With regard to the security achieved, firstly we achieve robustness in our model (security against denial of service attacks by a minority of corrupted parties), whereas no protocol can achieve robustness in the model of [12] where the attacker can simply block all communications (in their model, [12] achieve instead only the ‘agreement’ guarantee that all accepting honest players compute the same session key). Also, our model achieves security against ‘key malleability’ attacks by insiders which bias the key, whereas the model in [12] does not prevent such attacks. On the other hand, unlike our model, the model in [12] addresses ‘impersonation’ insider attacks (which arise only in their insider model, when the protocol views of different players differ), and is analyzed within the modular Universal Composability (UC) framework of Canetti [8].

Our protocol is an adaptation of existing techniques used for distributed key-generation in threshold cryptography [11]. Indeed, we use the same approach based on Pedersen’s verifiable secret sharing [15] to achieve robustness and prevent biasing of the key. However, the authors of [11] assume both a reliable broadcast channel as well as a private channel between shareholders, whereas we only assume a reliable broadcast channel, and use public-key encryption to implement a ‘virtual private channel’ over the broadcast channel. The adaptive chosen-ciphertext security (i.e. non-malleability) of the public-key encryption scheme is essential in proving the security against malleability attacks on the protocol, in which the insiders try to adapt their messages based on those sent by honest players.

2 Preliminaries

2.1 Covert Channel

Lampson [14, p. 614] informally specified a special type of communication being:

Covert channels, i.e. those not intended for information transfer at all, such as the service program’s effect on the system load.

In our context, the concept of subliminal channel, introduced in the limited context of authentication by Simmons [17] is more important. Insiders in a protocol can for example select randomness such that its (e.g. least significant) bit(s) reveal private data to an outsider.

2.2 Cryptographic Primitives and Security Notions

2.2.1 Discrete-Log Problem [10]. Let GC denote an algorithm that on input a security parameter k_s , returns an instance (D_G, g) of a multiplicative group G of prime order q with generator g (the description string D_G determines the group and contains the group order q). The discrete-log problem DL is defined as follows: Given $(D_G, g) = \text{GC}(k_s)$ and $y_1 = g^{x_1}$ for uniformly random $x_1 \in \mathbb{Z}_q$, compute x_1 . We say that DL is *hard* if the success probability $\text{Succ}_{\text{A,DL}}(k_s)$ of any efficient DL algorithm A with run-time $t(k_s)$ is upper-bounded by a negligible function $\text{InSec}_{\text{DL}}(t)$ of k_s .

2.2.2 Public-Key Encryption Schemes. Let $ES = (GKE, E, D)$ denote a public-key encryption scheme. Algorithm notation is as follows. On input security parameter k_s , $GKE(k_s)$ returns a secret/public key pair (sk, pk) . On input a public key pk and a message m , $E(pk, m)$ returns a ciphertext c . On input a secret key sk and a ciphertext c , $D(sk, c)$ returns a message m or an ‘invalid ciphertext’ symbol Rej . We require that ES satisfies the standard security notion INDCCA2 of indistinguishability under adaptive chosen-ciphertext attack [1], which is defined by the following game. The attacker $A = (A_1, A_2)$ runs in two stages, *find* and *guess*. In the *find* stage, a key-pair $(sk, pk) = GKE(k_s)$ is generated and A_1 is run on input pk_1 , and given access to the decryption oracle $D(sk, \cdot)$. At the end of the *find* stage, A_1 outputs a pair of messages (m_0, m_1) and a state string s . A random bit $b \in \{0, 1\}$ is chosen and a challenge ciphertext $c = E(pk, m_b)$ is computed. In the *guess* stage, A_2 is run on input (c, s) and given access to the decryption oracle $D(sk, \cdot)$, with the restriction that A_2 is disallowed from querying c to this oracle. At the end of the *guess* stage, A_2 outputs a bit $b' \in \{0, 1\}$. We say A *wins* the INDCCA2 game if $b' = b$, and denote this event by $Succ$. We define A ’s *advantage* against scheme ES in the sense of INDCCA2 by $Succ_{A, ES}^{INDCCA2}(k_s) = 2|\Pr[Succ] - \frac{1}{2}|$. We say that ES is secure in the sense of adaptive chosen-ciphertext attack (INDCCA2) if the advantage of any efficient attacker A with run-time t and q_d decryption queries is upper-bounded by a negligible *insecurity* function $InSec_{ES}^{INDCCA2}(t, q_d)$ of k_s . We say that ES is secure in the sense of chosen-plaintext attack (INDCPA) if the advantage of any efficient attacker A with run-time t and 0 decryption queries is upper-bounded by a negligible *insecurity* function $InSec_{ES}^{INDCPA}(t)$ of k_s .

3 Model

3.1 Background

As outlined in the introduction, our main goal for introducing the *shielded-insider privacy* security requirement is to prevent a minority of corrupted group insiders from biasing the session key, for example by fixing it to a value agreed beforehand with an outsider, so that the outsider can get some information on the session key. To give a better understanding of the issues involved, we first point out in this subsection several types of attacks on existing protocols which any good definition of *shielded-insider privacy* must exclude, and some attacks that we do not address. Then we present our definition.

‘Direct’ Malleability Attacks. The basic idea of ‘direct’ malleability attacks is as follows. When the protocol begins, the corrupted group players wait for the honest players to send messages containing their intended contribution to the session key. The corrupted players then use this information to compute their messages adaptively as some function of the honest player messages, in order to ‘cancel’ the contributions of the honest players and fix the session key as desired. It is important to remark that such attacks may be *undetectable* by the honest players, in the sense that all honest honest players compute the same key. An

attack of this type on the Burmester-Desmedt (BD) key exchange protocol [7] (as well as similar attacks on other protocols) was given by Pieprzyk and Wang [16] (see Appendix for details). The attack shows how a subset of only two corrupted players out of n can force the key computed at the end of the protocol by the honest players to any desired value. The attack applies also to the Katz-Yung protocol [13] which is based on the BD protocol.

The ‘Halting Attack’. A more subtle type of attack than the above ‘direct’ attacks, but still aimed at biasing the session key, was first described by Cleve [9] in the context of distributed coin tossing algorithms (and later revisited by Genaro, Jarecki, Krawczyk and Rabin [11]). In this type of attack, which we call a ‘halting attack’, the corrupted players wait until the honest players send their contributions as above. At that point in the protocol, the corrupted players can compute the value of session key (or at least some partial information on the key) that would result *if they continue to honestly follow the protocol*. If this value matches the desired bias for the key, the corrupted players continue honestly, but if it does not, the corrupted players simply *halt* (refuse to continue). If the protocol is not well designed, this may cause a new session key value to be computed (e.g. if the protocol is restarted). In this way the corrupted players can bias the session key of completed protocol instances towards their desired form, simply by deciding whether to halt or not based on the value of the key. Cleve proved [9] that *any* protocol for distributed coin-tossing in which the number of corrupted players is equal to or greater than the number of honest players, can be biased by a non-negligible amount using a halting attack. Therefore, the requirement for a honest majority in our protocol, which is a special type of distributed coin-tossing protocol, is optimal in order to defend against this type of attack.

The ‘Cloning Attack’. Besides the above attacks which attempt to control or bias the key, one can also consider ‘unshielded insider’ attacks which ‘leak’ the session key to an outsider who is able to eavesdrop on the protocol messages. Our ‘shielded insider’ model does *not* address such attacks — the ‘shielding’ requirement in our ‘shielded insider’ model means that the outsider cannot observe the protocol messages nor communicate with the insiders once the protocol session begins. We now motivate the need for this.

In fact, as we explain below, we believe that achieving resistance against ‘unshielded insider’ attacks is a problem related to elimination of ‘subliminal channels.’ Firstly, it is clear that an ‘unshielded’ insider can directly leak the session key to an eavesdropping outsider by broadcasting the session key in one of the insider’s ‘corrupted’ messages. But there are other more subtle ways in which the session key can be learned by an eavesdropping outsider *even if the insiders follow the protocol*. In a ‘cloning attack’, the eavesdropping outsider follows exactly the same steps as the corrupted insider (it is a program clone), using a sequence of random coins which was agreed upon before the protocol between the insider and outsider. Thus even if the insider follows the protocol, the outsider observing the messages received by the insider recovers the same session key as the insider. To avoid such generic attacks one may assume that

each player is provided with a true random tape, and that there is a ‘public key distribution’ initial stage in which a public function of each player’s random tape is broadcast to all players (the corrupted players are not allowed to temper with this initial broadcast). But even in this case we do not know how to eliminate subliminal channels which arise due to the use of a probabilistic encryption scheme in the protocol. So, addressing Katz-Yung’s open problem is far from trivial. Practical solutions may need to restrict the attacks from active insiders.

3.2 Definition of ‘Shielded-Insider’ Privacy and Robustness

Informal Discussion. Informally, we want the shielded-insider model to capture security against a group of up to k corrupted insiders collaborating with an outsider. The insiders can communicate with the outsider, but only until the protocol session begins, after which the insiders are shielded, i.e. prevented from further communication with the outsider. The insiders can then initiate several protocol sessions to exchange session keys with honest players. The insiders can try to choose their protocol messages in order to bias or correlate the session keys computed by honest players. To defeat the privacy, the outsider’s goal is then to distinguish these honest player session keys from independent random session keys. To defeat the robustness, the insider’s goal is to cause some honest players to compute different session keys from those computed by other honest players participating in the session.

Attack Model. The attacker A is modelled by a pair (A_I, A_O) of probabilistic algorithms (Turing machines) representing the collaborating insiders and outsiders, respectively. The attack model has three stages and runs as follows on input a security parameter k_s .

Stage 1 (Initialization): A set of common parameters $cp = \text{GCProt}(k_s)$ is generated using an algorithm GCProt . Long-Lived (LL) secret/public key-pairs $(sk_i, pk_i) = \text{GK}(k)$ (for $i = 1, \dots, \ell$) are generated for all potentially honest players in $P_{in} = \{U_1, \dots, U_\ell\}$ using the protocol LL key generation algorithm GK . The common parameters cp and vector of public keys $PK = (pk_1, \dots, pk_\ell)$ are now given to all players.

Stage 2 (Protocol Executions): The insider attacker A_I is run on input (cp, PK) and initiates exactly q_{ex} protocol sessions (see `Execute` oracle below). At any instant of time in this stage, we let P denote the set of potential participating players in protocol executions. The set P is initialized to the set $P_{in} = \{U_1, \dots, U_\ell\}$ of honest players but grows as A_I corrupted players to P (see `AddPlayer` oracle below). In this stage A_I is given access to the following oracles (we emphasize that the `Execute` and `Send` oracles are different from the corresponding oracles in the K-Y model). Note that below, when we say that a message is “broadcast” by some party, we mean that the message is reliably delivered to all parties in the corresponding session (honest players and A_I):

1. **Execute(S):** Starts a new protocol session among n unused instances of players specified by the set $S = \{U_{i[1]}, \dots, U_{i[n]}\}$, where each $U_{i[j]} \in P$. The set S must contain at most k corrupted players. The ‘start session’ message (init, S) is broadcast to all honest players in S . The round clock for this session is started an ‘end of round’ clock message (endrnd, i) is broadcast at the end of the i th round. The duration T_i of the i th round is fixed in advance as a function $T_i(k_s)$ of the security parameter. Thus each session started by an **Execute** query terminates after time $T = \sum_i T_i$. At the end of each round, the honest players process all the messages received during this round and broadcast their messages for the next round (or terminate after the last round). Then A_1 broadcasts up to μ messages for this round on behalf of either corrupted players or honest players (see **Send** oracle below).
2. **Send(s, M):** Broadcasts message M to all parties in session s . The message is added by each honest player instance to its receive buffer for the current round (to be processed at the end of the round). The message M may be of two types: (1) A *corr* message is a message on behalf of a corrupted player (such messages have sender prefix U_i for some corrupted player U_i) or (2) A *spoof* message is a message on behalf of an honest player (such messages have sender prefix U_i for some honest player U_i). Note that in the latter case all players will receive at least two messages with sender prefix U_i (one correct and one spoofed). We allow A_1 up to μ messages for each session’s round.
3. **AddPlayer(U, pk_U):** Adds new corrupted player U to potential player set P with public-key pk_U , which becomes available to all honest players in P_{in} . Protocol sessions containing U as a partner can subsequently be started by A_1 .
4. **OutComm(M):** Sends message M to outsider attacker A_O and outputs A_O ’s response message. Note that this query is allowed only before A_1 ’s first **Execute** query (“shielding” requirement).
 Stage 2 ends when the last protocol session (number q_{ex}) initiated by A_1 ends.

Stage 3 (Distinguish from Random): For $i = 1, \dots, q_{ex}$, let S_i denote A_1 ’s **Execute** query for initiating the i th protocol session. We let $sk_1[i]$ denote the session-key computed at the end of this session by the first honest player instance in S_i , and we let $sk_0[i]$ denote a key chosen independently at random from the session key space SP_{sk} , according to the session-key probability distribution output by honest protocol executions. An independent random bit $b \in \{0, 1\}$ is chosen and the outsider A_O is sent a message ($\text{Test}, (sk_b[1], \dots, sk_b[q_{ex}])$). Finally, A_O outputs an estimate b' for the bit b .

Shielded-Insider Privacy Security Notion. We say $A = (A_1, A_O)$ *wins* in the sense of breaking ‘shielded-insider privacy’ (sh-pr notion) if A_O ’s estimate of b was correct ($b' = b$). If Succ_{pr} denotes the event that A wins then we define A ’s advantage against protocol Prot in the shielder-insider notion sh – ins by $\text{Succ}_{A, \text{Prot}}^{\text{sh-pr}}(k) \stackrel{\text{def}}{=} 2(\text{Pr}[\text{Succ}_{pr}] - 1/2)$. We define the insecurity

$\text{InSec}_{\text{Prot}}^{\text{sh-pr}}(t, q_{ex}, q_s, \ell_B)$ of Prot in the sense of sh-pr against attackers A with resource parameters: run-time t , q_{ex} Execute queries, q_s Send queries, and Send query length bound ℓ_B , as the maximal advantage $\text{Succ}_{A, \text{Prot}}^{\text{sh-pr}}(k)$ of any attacker A with these resource parameters. We say Prot achieves shielded-insider privacy if $\text{InSec}_{\text{Prot}}^{\text{sh-pr}}(t, q_{ex}, q_s, \ell_B)$ is a negligible function of the security parameter k_s whenever the resource parameters are polynomial in k_s .

Robustness Security Notion. We say that A_I wins in the sense of breaking ‘robustness’ (rob notion) if at the end of Stage 2, there exists a pair of honest player who were partners in the same protocol session, but computed different session keys (or one or both of them aborted). We denote A_I ’s winning probability in this game by $\text{Succ}_{A, \text{Prot}}^{\text{rob}}(k)$, and define the insecurity $\text{InSec}_{\text{Prot}}^{\text{rob}}(t, q_{ex}, q_s, \ell_B)$ of Prot in the sense of rob against attackers with resource parameters (t, q_{ex}, q_s, ℓ_B) (defined above) as the maximal success probability $\text{Succ}_{A, \text{Prot}}^{\text{rob}}(k)$ of any attacker with these resource parameters. We say Prot achieves robustness if $\text{InSec}_{\text{Prot}}^{\text{rob}}(t, q_{ex}, q_s, \ell_B)$ is a negligible function of the security parameter k_s whenever the resource parameters are polynomial in k_s .

Privacy and Robustness Against Non-spoofing (ns) Attackers. To simplify our analysis, we will first present a basic protocol which achieves security against a weaker attacker than described above, called a *non-spoofing attacker*. Then we will show how to strengthen the protocol to be secure against the more general attackers above. A *non-spoofing* (ns) attacker is restricted to make no *spoo*f queries to the Send oracle, i.e. it never sends any messages on behalf of honest players. We call the corresponding shielded-insider privacy and robustness notions ns-sh-ins and ns-rob, respectively.

3.3 Definition of Key Privacy Against Outsiders

Informal Discussion. To define security against outsiders, we use the Katz-Yung model [13]. Note that in one sense, the attacker’s capability is weaker in this outsider model compared to the insider model of the previous section, because here all protocol players are assumed to be honest (compared with up to k corrupted players in the insider model). However in another sense, the attacker’s capability in this model is stronger because here the attacker has full control over the network, and hence can modify or block any message sent by an honest player (whereas in the insider model a reliable broadcast channel was available to honest players). Also, in this model the attacker is allowed to view all the protocol messages in order to help in distinguishing the final key from a random key (whereas in the insider model, the outsider cannot see the protocol messages — note that the insider who sees the protocol messages can trivially distinguish the key from random by recovering it).

For completeness, we now recall this model [13].

Attack Model. The attacker A is modelled by a probabilistic algorithm (Turing machine) representing the outsider. The attack model has three stages and runs as follows on input a security parameter k_s .

Stage 1 (Initialization): A set of common parameters $cp = \text{GCProt}(k_s)$ is generated using an algorithm GCProt . Long-Lived (LL) secret/public key-pairs $(sk_{U_i}, pk_{U_i}) = \text{GK}(k)$ (for $i = 1, \dots, \ell$) are generated for all potentially honest players in $P_{in} = \{U_1, \dots, U_\ell\}$ using the protocol LL key generation algorithm GK . The common parameters cp and vector of public keys $PK = (pk_{U_1}, \dots, pk_{U_\ell})$ are now given to all players.

Stage 2 (Protocol Executions): Each player $U \in P_{in}$ is allowed an unlimited number of *instances*, each executing a session of the protocol. The i th instance of player U is denoted by Π_U^i , and may be used only for one session. Each instance Π_U^i maintains a set of variables state_U^i , term_U^i , acc_U^i , used_U^i , sid_U^i , pid_U^i and sk_U^i , as used in [2]. The goal of the protocol is for an instance Π_U^i to end up in an *accepting* state ($\text{acc}_U^i = \text{TRUE}$) with the session key stored in sk_U^i . The attacker A has complete control over the network - this is modelled by giving A access to the following *oracles*:

1. $\text{Send}(U, i, M)$ – Sends the message M to instance Π_U^i and returns Π_U^i 's response to A . The message M has two possible forms: (1) Send_0 query: This message has the form $M = (\text{Init}, S)$, where $S = \{U_{i[1]}, \dots, U_{i[n]}\}$, where each $U_{i[j]} \in P_{in}$. This is the first message sent to instance Π_U^i to prompt it to start a session with the players in set S . (2) All other Send queries have the form $M = (U_1|j|m_1, U_2|j|m_2, \dots, U_n|j|m_n)$, where $U_i|j|m_i$ is a j th round broadcast from sender U_i , with ‘payload’ message m_i . The instance Π_U^i 's response always has the form $U|j|m$. *Note:* To be consistent with the ‘spoofer’ queries in the insider model, we actually allow Send queries with $M = (S_1, \dots, S_n)$, where $S_i = \{U_i|j|m_{i,1}, \dots, U_i|j|m_{i,\beta}\}$ is a *set* of β received broadcasts with sender prefix U_i .
2. $\text{Execute}(S)$: Executes a protocol session (with reliable network) among n unused instances of players specified by set $S = \{U_{i[1]}, \dots, U_{i[n]}\}$, where each $U_{i[j]} \in P_{in}$, and returns the complete protocol transcript to A .
3. $\text{Reveal}(U, i)$: Returns the session key sk_U^i to A .
4. $\text{Corrupt}(U)$: Returns player U 's LL key sk_U to A .
5. $\text{Test}(U, i)$: This query is allowed just once during the attack. A random bit b is generated. If $b = 1$, the oracle returns the session key sk_U^i , otherwise if $b = 0$ it returns an independent random session key.

An *active* attacker has access to all of the above oracles. A *passive* attacker has access to all oracles except the Send oracle. Below we define a third type of attacker called a *Block-or-Forward* (BoF) attacker, which is intermediate between a passive and an active attacker.

Stage 3 (Distinguish from Random): Eventually A returns an output bit b' (which is A 's estimate of the bit b).

Partnering. The *session ID* sid_U^i for instance Π_U^i is a protocol-specified function of all communication sent and received by Π_U^i . The *partner ID* pid_U^i for instance Π_U^i consists of the set of identities of players with whom Π_U^i intends

to communicate, as defined by the set S contained in the Send_0 query sent to Π_U^i . Two instances Π_U^i and $\Pi_{U'}^{i'}$ are *partnered* if both (1) $\text{sid}_U^i = \text{sid}_{U'}^{i'}$, and (2) $\text{pid}_U^i = \text{pid}_{U'}^{i'}$ hold.

Freshness. An instance Π_U^i is *fresh* unless one of the following *bad* events occurred (here $\Pi_{U'}^{i'}$ is any instance partnered with Π_U^i): (1) A queried $\text{Reveal}(U, i)$ or $\text{Reveal}(U', i')$, (2) A queried $\text{Corrupt}(V)$ for some $V \in \text{pid}_U^i$ and then made a Send query to either Π_U^i or $\Pi_{U'}^{i'}$.

Outsider Security Notion. We say $A = (A_I, A_O)$ *wins* in the sense of breaking ‘outsider privacy’ (AKE notion) if A queried Test on a fresh instance Π_U^i which has accepted (i.e. $\text{acc}_U^i = \text{TRUE}$) and A correctly guessed the Test oracle’s bit b (i.e. $b' = b$). Let Succ denotes the event that A wins. Then we define A’s advantage against protocol Prot by $\text{Succ}_{A, \text{Prot}}^{\text{AKE}}(k) \stackrel{\text{def}}{=} 2|\Pr[\text{Succ}] - 1/2|$. We define the insecurity $\text{InSec}_{\text{Prot}}^{\text{AKE}}(t, q_{ex}, q_s, \ell_B)$ of Prot in the sense of AKE against attackers A with resource parameters (t, q_{ex}, q_s, ℓ_B) (run-time t , q_{ex} Execute queries, q_s Send queries, and ℓ_B Send query length bound), as the maximal advantage $\text{Succ}_{A, \text{Prot}}^{\text{AKE}}(k)$ of any attacker A with these resource parameters. We say Prot achieves shielded-insider privacy if $\text{InSec}_{\text{Prot}}^{\text{AKE}}(t, q_{ex}, q_s, \ell_B)$ is a negligible function of the security parameter k_s whenever the resource parameters (t, q_{ex}, q_s, ℓ_B) are polynomial in k_s .

Privacy and Robustness Against “Block-or-Forward” (BoF) Attackers. As mentioned above, we use a modular approach similar to that used by Katz and Yung [13]. First we present a basic protocol which achieves security against a weaker outsider attacker than the active attacker described above, called a *Block-or-Forward* (BoF) attacker. Then we will show a generic conversion to strengthen any protocol secure against BoF attackers to be secure against active attackers. A *BoF* attacker is a restricted type of active attacker. We impose the following restrictions on a BoF attacker A:

1. A is allowed to initiate at most one instance Π_U^1 of each player U (i.e. at most one Send_0 query to each player).
2. A is allowed to either block or forward (but not modify) messages sent by players. More precisely, each Send query of A to instance Π_U^1 , except the Send_0 query, has the form $\text{Send}(U, 1, M)$ with $M = (U_1|j|m_1, U_2|j|m_2, \dots, U_n|j|m_n)$, such that for each $l = 1, \dots, n$, either (a) “Forward” case: $U_l|j|m_l$ was a response to a previous Send query of A to $\Pi_{U_l}^1$, or (b) “Block” case: m_l is the ‘empty’ string.
3. A makes at most $q_{ex} = 1$ Execute query, and makes no Reveal queries.

Note that we also slightly relax the definition of a BoF attacker A *winning* (compared to the definition for active attackers) in that we do not require the instance that A queries Test on to be fresh (so a BoF attacker can win even if he queries Corrupt to all players). Otherwise, the definitions are identical, and

we denote the corresponding security notion by KE-BoF and the corresponding insecurity function of Prot against BoF attackers by $\text{InSec}_{\text{Prot}}^{\text{KE-BoF}}(t, q_{ex}, q_s, \ell_B)$.

Remark. Any protocol which is secure against passive attacks can be easily converted into a protocol secure against BoF attacks *if we do not require that the protocol achieve insider security*. Namely, one can modify the original protocol to simply abort any player who does not receive a message (due to blocking). But this method clearly does *not* apply if we also want the protocol to achieve insider robustness.

4 The Basic Protocol

In this section we give a basic version of our protocol which is secure against non-spoofing attacks (see Section 3). Later we upgrade it to achieve the full security notions.

Cryptographic Primitives. Our basic protocol makes use of the following cryptographic primitives (see Section 2.2 for notation and definitions) :

- (1) **Public Key Encryption Scheme** $\text{ES} = (\text{GKE}, \text{E}, \text{D})$. We require that ES satisfies the standard security notion IND-CCA2 of indistinguishability under adaptive chosen-ciphertext attack [1]
- (2) **Group G of Prime Order q** . We require that the discrete-log problem DL is hard in G . We let $l_q = \log_2(q)$.

The Protocol. Our protocol incorporates the Pedersen Verifiable Secret Sharing (VSS) scheme [15] over the group G . The Pedersen VSS scheme is also used in a similar way in the distributed discrete-log threshold key generation scheme in [11]. The protocol has a security parameter $k_s \in \mathbb{N}$ and a parameter k specifying the number of corrupted insiders that can be tolerated.

Let $P_{in} = (U_1, \dots, U_\ell)$ denote the set of all $\ell \geq 2k + 1$ potential players in future protocol instances (any subset of P_{in} may later run an instance of the protocol). Before any session keys can be exchanged, the following initialization phase is run.

Initialization. A set of common parameters is generated for the Pedersen VSS scheme: An instance of group G of prime order q and generator $g \in G$ is generated with $(D_G, g) = \text{GC}(k_s)$. A random $x \xleftarrow{\text{R}} \mathbb{Z}_q$ is chosen and we set $h = g^x$. The common parameters are (D_G, g, h) , where D_G contains q (We let $l_q = \log_2(q)$). There are no long-lived keys.

Protocol Instance. When a subset P of $n \geq 2k + 1$ players from P_{in} wish to generate a common session key, maintaining robustness against up to k corrupted players, they use the following protocol (to simplify notation in the following we assume these n players to be $P = \{U_1, \dots, U_n\}$ — the n players can always be ordered lexicographically by identity).

1. Encryption Key Distribution:

- (a) **Send.** Each player U_i generates a random encryption key-pair $(sk_i, pk_i) = \text{GKE}(k_s)$ and broadcasts message $U_i|1|pk_i$.
- (b) **Receive.** Each player U_i receives a vector of n encryption keys (pk_1, \dots, pk_n) , where pk_j denotes the key received from U_j .

2. Pedersen $(k + 1, n)$ VSS Subkey Commitment by $k + 1$ players:

- (a) **Send.** Each player $U_i \in \{U_1, \dots, U_{k+1}\}$ chooses a uniformly random subkey $K_i \in \mathbb{Z}_q$ and computes n shares $(y_i[1], s_i[1]), \dots, (y_i[n], s_i[n])$ of K_i along with a commitment vector $\mathbf{v}_i = (v_i[0], \dots, v_i[k])$ using a $(k + 1, n)$ Pedersen VSS scheme: U_i chooses two uniformly random polynomials $f_i(z)$ and $g_i(z)$ of degree k over \mathbb{Z}_q , where:

$$f_i(z) = K_i + x_i[1] \cdot z + \dots + x_i[k] \cdot z^k \text{ and } g_i(z) = R_i + r_i[1] \cdot z + \dots + r_i[k] \cdot z^k.$$

Then for $j = 1, \dots, n$, the j th share is $(y_i[j], s_i[j]) = (f_i(j) \bmod q, g_i(j) \bmod q)$, and U_i appends the sender prefix U_i and encrypts it using U_j 's key to get $c_i[j] = \text{E}(pk_j, U_i|(y_i[j], s_i[j]))$. For $j = 1, \dots, k$, the j th commitment is $v_i[j] = g^{x_i[j]}h^{r_i[j]}$, and the zero'th commitment is $v_i[0] = g^{K_i}h^{R_i}$. U_i broadcasts the message $U_i|2|(\mathbf{v}_i, c_i[1], \dots, c_i[n])$.

- (b) **Receive.** Each player U_i receives a message $U_j|2|(\mathbf{v}_j, c_j[1], \dots, c_j[n])$ from each player U_j for $j = 1, \dots, k$, decrypts $c_j[i]$ to get share $sp|(y_j[i], s_j[i]) = \text{D}(sk_i, c_j[i])$ with sender prefix sp , and checks that $sp = U_j$ and the share validity relation $g^{y_j[i]}h^{s_j[i]} = \prod_{l=0}^k v_j[l]^{j^l}$ holds: if either check fails, U_i adds U_j to a complaint set Comp_i .

3. Broadcast Complaints:

- (a) **Send.** Each player U_i broadcasts the message $U_i|3|\text{Comp}_i$.
- (b) **Receive.** Each player U_i receives a message $U_j|3|\text{Comp}_j$ from each player U_j . If U_i belongs to $k + 1$ or more received complaint sets Comp_j then U_i aborts the protocol.

4. Broadcast Complaint Responses and Compute Qualified Set $QUAL_i$:

- (a) **Send.** Each player $U_i \in \{U_1, \dots, U_{k+1}\}$ responds to each complaint set Comp_j which contains U_i by revealing the j th share $(y_i[j], s_i[j])$: U_i broadcasts the message $U_i|4|\text{CompR}_i$, where $\text{CompR}_i = \{(j, (y_i[j], s_i[j]))\}_{j:U_i \in \text{Comp}_j}$.
- (b) **Receive.** Each player U_i receives a message $U_j|4|\text{CompR}_j$ from each player U_j . Now U_i computes the player set $QUAL_i$ as follows. U_i initializes $QUAL_i$ to $\{U_1, \dots, U_{k+1}\}$. For each complaint set Comp_l which contains U_j (for $l = 1, \dots, n$), U_i checks that U_j responded correctly by checking if CompR_j contains a share $(l, (y_j[l], s_j[l]))$ which satisfies the share validity test $g^{y_j[l]}h^{s_j[l]} = \prod_{m=0}^k v_j[m]^{l^m}$: if the check fails, or if U_j belongs to $k + 1$ or more complaint sets Comp_j , then U_i removes U_j from $QUAL_i$. Note that by definition, U_i now holds a valid i th share $(y_j[i], s_j[i])$ for each $U_j \in QUAL_i$.

5. SubKey Decommitment:

- (a) **Send.** Each player $U_i \in \{U_1, \dots, U_{k+1}\}$ encrypts a decommitment (K_i, R_i) (with appended sender prefix U_i) to his sub-key using U_j 's public key to get $c'_i[j] = \mathbf{E}(pk_j, U_i|(K_i, R_i))$, for $j = 1, \dots, n$. U_i broadcasts the message $U_i|5|(c'_i[1], \dots, c'_i[n])$.
- (b) **Receive.** Each player U_i receives a message $U_j|5|(c'_j[1], \dots, c'_j[n])$ from each player U_j for $j = 1, \dots, k$, decrypts $c_j[i]$ to get decommitment $sp|(K_j, R_j) = \mathbf{D}(sk_i, c_j[i])$ with sender prefix sp , and checks that $sp = U_j$ and the decommitment validity relation $g^{K_j} h^{R_j} = v_j[0]$ holds: if either check fails, U_i adds U_j to a complaint set Comp'_i , otherwise U_i holds a valid decommitment (K_j, R_j) .

6. Broadcast Decommitment Complaints:

- (a) **Send.** Each player U_i broadcasts the message $U_i|6|\text{Comp}'_i$.
- (b) **Receive.** Each player U_i receives a message $U_j|6|\text{Comp}'_j$ from each player U_j .

7. Broadcast Complaint Responses and Recover Session Key:

- (a) **Send.** Each player U_i responds to a complaint on U_j by encrypting U_j 's i -th VSS share $(y_j[i], s_j[i])$ to each player: U_i computes the ciphertext vectors $\mathbf{c}''_{i,j} = (c''_{i,j}[1], \dots, c''_{i,j}[n])$, where $c''_{i,j}[l] = \mathbf{E}(pk_l, U_i|(y_j[i], s_j[i]))$ for $l = 1, \dots, n$. U_i broadcasts the message $U_i|7|\{(U_j, \mathbf{c}''_{i,j})\}_{U_j \in \text{Comp}'}$, where Comp' is the union of $\text{Comp}'_1, \dots, \text{Comp}'_n$ (the set of all players complained about in previous round).
- (b) **Receive.** Each player U_i receives a message $U_l|7|\{(U_j, \mathbf{c}''_{l,j})\}_{U_j \in \text{Comp}'}$ from each player U_l . For each $U_j \in \text{QUAL}_i$ which is also in Comp'_i , U_i recovers (K_j, R_j) by decrypting the ciphertexts containing U_j shares: U_i computes $sp|(y_j[l], s_j[l]) = \mathbf{D}(sk_i, c''_{l,j}[i])$ for $l = 1, \dots, n$. Using $k + 1$ such valid shares with $l \in S^*$ for some subset $S^* \subseteq \{1, \dots, n\}$ (satisfying the VSS relations $g^{y_j[l]} \cdot h^{s_j[l]} = \prod_{t=0}^k v_j[t]^{l^t}$, and having valid sender prefix $sp = U_l$), U_i reconstructs (K_j, R_j) by Lagrange interpolation:

$$K_j = \sum_{l \in S^*} \lambda_{S^*,l} \cdot y_j[l] \bmod q \text{ and } R_j = \sum_{l \in S^*} \lambda_{S^*,l} \cdot s_j[l] \bmod q,$$

where $\lambda_{S^*,l}$ is the l th Lagrange coefficient with respect to S^* . Now U_i holds K_j for all $U_j \in \text{QUAL}_i$ (otherwise U_i aborts the protocol), computes the session key $SK_i = \sum_{j \in \text{QUAL}_i} K_j \bmod q$, and terminates with success.

Note: At the end of each (say j th) round, each player U_i is receives a message from each player U_l of the form $U_l|j|m$. However, it may happen due to attacker behaviour, that U_i either (1) receives no messages from U_l or (2) more than one message from U_l . In case (1), all validity checks specified above involving the received message are defined to have failed. In case (2), we assume the receiver picks an arbitrary single message from those that were received from U_l and proceeds as described above.

Efficiency Remarks

- (1) Our basic protocol has 7 rounds. The dominant computation/communication costs in the ‘optimistic case’ where all players follow the protocol are the VSS computations in rounds 2, 5 and 7 (namely $O(k)$ exponentiations in G and $O(n)$ multiplications modulo q by $k + 1$ sending players, and $O(1)$ exponentiations and $O(kn \log n)$ group operations in G by n receiving players) and the encryption computations in rounds 2 and 5 (namely $O(n)$ ciphertexts encrypted/sent by first $k + 1$ players and $O(k)$ ciphertexts decrypted by n players). In the worst case of k corrupted players, round 7 requires also $O(n)$ encryptions/decryptions of messages/ciphertexts of length $O(nk)$ per player.
- (2) When n is much bigger than k , one can improve the efficiency by running the first 7 rounds among the first $2k + 1$ players only, and then distributing ciphertexts of the resulting session key K (together with the decommitment randomness R to allow verification) to the remaining $n - (2k + 1)$ players.

5 Security Proofs for the Basic Protocol

The insider security of the basic protocol is summarised by the following statement (see full version of the paper for proof).

Theorem 1 (Security Against Non-Spoofing Insiders). (1) *If the Discrete-Log problem (DL) is hard in the group G generated by the common-parameter algorithm GC, then protocol Prot achieves robustness against non-spoofing insider attacks (ns-rob notion). (2) If in addition, the encryption scheme ES is secure under adaptive chosen-ciphertext attack (INDCCA2 notion), then protocol Prot achieves shielded-insider privacy against non-spoofing insider attack by k insiders (ns-sh-pr notion). Concretely, the following insecurity bounds hold:*

$$\begin{aligned} \mathbf{InSec}_{\text{Prot}}^{\text{ns-rob}}(t, q_{ex}, q_s, \ell_B) &\leq \mathbf{InSec}_{\text{DL}}(t[D]), \\ \mathbf{InSec}_{\text{Prot}}^{\text{ns-sh-pr}}(t, q_{ex}, q_s, \ell_B) &\leq 2[\mathbf{InSec}_{\text{DL}}(t[D]) \\ &\quad + q_{ex}(\ell - 1)\mathbf{InSec}_{\text{ES}}^{\text{INDCCA2}}(t[E], q_d[E])], \end{aligned}$$

where $t[D] = t + q_{ex}O((k+1)^2l_q^2 + (k+1) \cdot \ell \cdot l_q)$, $t[E] = t + O(\ell^2k^3(l_c + l_{pk} + l_{ID} + l_q))$, $q_d[E] = k(k+2)$. Here l_q is an upper bound on the group order q and l_c , l_{ID} and l_{pk} denote upper bounds on the length of ciphertexts sent by corrupted players, length of player identities and length of public keys for encryption scheme ES, respectively.

The outsider security of the basic protocol is summarised as follows.

Theorem 2 (Security Against BoF Outsiders). *If the encryption scheme ES is secure under chosen-plaintext attack (INDCPA notion), then protocol Prot achieves outsider privacy against Block-or-Forward attackers (KE-BoF notion). Concretely, the following insecurity bounds hold:*

$$\text{InSec}_{\text{Prot}}^{\text{KE-BoF}}(t, q_s, \ell_B) \leq [\ell(\ell - 1)(k + 2) + 2(\ell - 1)(k + 1)] \text{InSec}_{\text{ES}}^{\text{INDCPA}}(t[E]),$$

where $t[E] = t + O(\ell^2 k(l_c + l_{pk} + l_{ID} + l_q))$. Here l_q is an upper bound on the group order q and l_c , l_{ID} and l_{pk} denote upper bounds on the length of ciphertexts sent by corrupted players, length of player identities and length of public keys for encryption scheme ES, respectively.

6 Adding Authentication to the Basic Protocol

In the full version of this paper, we describe a generic “compiler” KY' which takes any given input protocol Prot which is “weakly” secure (namely, achieves KE-BoF security against BoF attackers in the outsider model and sh-pr-ns and rob-ns security against non-spoofing attackers in the insider models) and outputs a “strengthened” protocol Prot' which is “strongly” secure (namely, achieves KE security against active attackers in the outside model and sh-pr and rob security in the insider models). Applying this compiler to our basic protocol Prot from Section 4 gives our full authenticated protocol. Due to space limitations we could not include this result in this version of the paper.

Our compiler KY' is a slight modification of the compiler KY described by Katz and Yung [13]. The modification is needed because the original compiler KY upgrades *outsider* security but does not upgrade the *insider* security — indeed, protocols output by the compiler KY are not robust in the insider model, since the KY compiler aborts any honest player who receive an incorrectly signed message (or zero messages or more than one message) from another player in some round. To fix this problem, our modified compiler KY' simply “filters out” (ignores) the incorrectly signed received messages, and if no correctly signed message arrives, passes the ‘empty’ string to the underlying input protocol player. Thanks to this modification and the signatures added by KY' , the output protocol Prot' of KY' can be shown secure in the insider models as long as the input protocol Prot is secure against non-spoofing attackers. On the other hand, in the outsider model, the modification means that the input protocol Prot must be secure against Block-or-Forward attackers in order for Prot' to be secure against active attackers. That is, outsider security of Prot against passive attackers does not suffice for the compiler KY' , whereas it was enough for the KY compiler.

Acknowledgements. The first author has been funded by NSF CCR-0209092, the others by the Australian Research Council grants DP0451484 and DP0663452. A part of this research was conducted when the first author visited Macquarie University in July 2003. This visit was supported by the Australian Research Council grant DP0344444.

References

1. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 26–45, Berlin, 1998. Springer-Verlag.

2. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155, Berlin, 2000. Springer-Verlag.
3. C. Boyd and J. Nieto. Round-Optimal Contributory Conference Key Agreement. In *PKC 2003*, volume 2567 of *LNCS*, pages 161–174, Berlin, 2003. Springer-Verlag.
4. E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange – The Dynamic Case. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 290–309, Berlin, 2001. Springer-Verlag.
5. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 321–336, Berlin, 2002. Springer-Verlag.
6. E. Bresson, O. Chevassut, D. Pointcheval, and J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Communications and Computer Security Conference CCS 2001*, pages 255–264, New York, 2001. ACM.
7. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *EUROCRYPT '94*, volume 950 of *LNCS*, Berlin, 1994. Springer-Verlag.
8. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005. Available from <http://eprint.iacr.org/>.
9. R. Cleve. Limits on the Security of Coin Flips When Half the Processors are Faulty. In *Proc. 18-th STOC*, pages 364–369, New York, 1986. ACM Press.
10. W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. on Information Theory*, 22:644–654, 1976.
11. Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *EUROCRYPT '99*, volume 1592 of *LNCS*, pages 295–310, Berlin, 1999. Springer-Verlag.
12. J. Katz and J. Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *Communications and Computer Security Conference CCS 2005*, pages 180–189, New York, 2005. ACM.
13. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125, Berlin, 2003. Springer-Verlag.
14. B. W. Lampson. A note on the confinement problem. *Comm. ACM*, 16(10):613–615, October 1973.
15. T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO '91*, volume 576 of *LNCS*, pages 129–140, Berlin, 1992. Springer-Verlag.
16. J. Pieprzyk and H. Wang. Key Control in Multi-party Key Agreement Protocols. In *Workshop on Coding, Cryptography and Combinatorics (CCC 2003)*, LNCS, Berlin, 2003. Springer-Verlag.
17. G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology. Proc. of Crypto 83*, pages 51–67. Plenum Press N.Y., 1984. Santa Barbara, California, August 1983.

A Malleability Attack on BD Protocol

For completeness, we describe the malleability attack [16] on the BD protocol mentioned in Section 3. First we recall the BD protocol [7]. Given a group G with generator g , the n players U_0, \dots, U'_{n-1} generate a session key as follows:

1. Each player U_i chooses random $r_i \in \mathbb{Z}_q$ and broadcasts $z_i = g^{r_i}$.
2. Each player U_i broadcasts $X_i = (z_{i+1}/z_{i-1})^{r_i}$.
3. Each player U_i computes their session key $K_i = z_{i-1}^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2}$.

The malleability attack by two corrupted players U_{n-2} and U_{n-1} to fix the session key to some value K_F proceeds as follows. In first round, the corrupted players follow the protocol honestly. In the second round they wait until the honest players ($i \in \{0, \dots, n-3\}$) broadcast their X_i values. Then U_{n-2} and U_{n-1} compute the session key K that would result if they proceed honestly, and broadcast the ‘bad’ values $X'_{n-2} = (K_F/K) \cdot X_{n-2}$ and $X'_{n-1} = (K/K_F) \cdot X_{n-1}$, respectively (here X_{n-2} and X_{n-1} are the values that U_{n-2} and U_{n-1} should have broadcasted if they wanted to follow the protocol). One can verify that each honest player U_i computes a session key

$$K_i = z_{i-1}^{nr_i} \cdots (X'_{n-2})^{i+1} \cdot (X'_{n-1})^i \cdots X_{i-2} = K_F$$

since $(X'_{n-2})^{i+1} \cdot (X'_{n-1})^i = (K_F/K)^{i+1} \cdot (K/K_F)^i \cdot X_{n-2}^{i+1} \cdot X_{n-1}^i = K_F/K \cdot X_{n-2}^{i+1} \cdot X_{n-1}^i$. Thus all honest players compute the key K_F forced by the corrupted players.

Formalising Receipt-Freeness

H.L. Jonker¹ and E.P. de Vink^{1,2}

¹ Dept. of Math. and Comp. Sc., Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

² LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract. Receipt-freeness is the property of voting protocols that a voter cannot create a receipt which proves how she voted. Since Benaloh and Tuinstra introduced this property, there has been a large amount of work devoted to the construction of receipt-free voting protocols. This paper provides a generic and uniform formalism that captures the notion of a receipt. The formalism is then applied to analyse the receipt-freeness of a number of voting protocols.

Keywords: online voting schemes, receipt-freeness, formal methods, eGovernment.

1 Introduction

In 2005, the Estonian government held elections for local office. Voters could cast their votes online or in a traditional voting booth. This constituted the first application of online voting on a national scale. Prior to the Estonian 2005 local office elections, a large amount of research has been conducted into voting and online voting. Voting is a means to establish consensus for a group of people regarding a certain set of candidates or choices. There are various methods to conduct an election, such as collecting one vote per voter (1V1V for short), e.g. elections for a governmental office, or collecting a list for each voter which orders the candidates by voter's preference (e.g. the scheme used to determine the Dutch *Top 2000* music list). This paper focuses upon 1V1V online voting protocols, in which all votes have equal weights.

Online voting schemes can offer an important advantage over traditional elections involving paper ballots in voting booths: voting is not restricted to a limited number of locations which have limited opening hours, but can be done anywhere a network connection is available, for as long as the elections are open. However, there is also a danger: strengths of the paper ballot system may be lost and new flaws can be introduced. To prevent this, electronic voting schemes should be designed to comply with a complete set of requirements.

Over the years, several desirable properties of electronic elections have been distinguished (see e.g. [15,10]). In this paper we focus on the property of *receipt-freeness*. A receipt allows a voter to prove how she voted. If receipts cannot be constructed, the voting protocol is said to be receipt-free.

Most other established properties of voting protocols, such as universal verifiability (any party can verify that the result is compromised of all legitimately

cast votes, and nothing else), democracy (only eligible voters can vote, and all voters can vote once), accuracy (no vote can be altered, duplicated or forged without being detected), robustness (the system can resist faulty behaviour of any reasonably sized coalition of participants) and integrity (the result of a vote consists precisely of all valid votes, and nothing more), are orthogonal to the concept of receipt-freeness. However, the notion of individual verifiability can be in conflict with it. Individual verifiability is the property that an individual voter can verify that her vote is correctly taken into account in computing the final tally. If this verification is transferable, the verification and the keys used to encrypt and/or blind the vote together would constitute a receipt. One method to reconcile the two properties is to use designated-verifier proofs [13].

This paper formalises the notion of receipt-freeness, which was introduced in the seminal work of Benaloh and Tuinstra [3]. In the early nineties, this property was violated by virtually all online voting schemes of that time. (Note that paper ballot 1V1V voting systems usually are receipt-free, which means that an online voting system which allows for receipts introduces a new flaw.) The lack of receipt-free schemes prompted the development of voting schemes, which would account for receipt-freeness in addition to other requirements.

In [3], the concept of receipt-freeness was introduced for a particular protocol (which we name BT). The definition does not lend itself for a convenient generalisation, however. It relies on distinguishing votes for candidate ‘0’ from votes for candidate ‘1’. The main contribution of our paper is a constructive, generic formalisation of the notion of receipt-freeness in a uniform framework for 1V1V voting schemes. The process algebraic setting adopted here facilitates reasoning about and verification of receipt-freeness and enables detection and identification of receipts. The proposed formalism is then applied to the voting protocols BT of Benaloh and Tuinstra [3], SK95 of Sako and Kilian [18], HS of Hirt and Sako [11], ALBD of Aditya et al. [1] (which are receipt-free) and the RIES protocol as discussed in [12] (which is not).

Since Benaloh and Tuinstra’s work there has been much research into receipt-freeness. In some works, e.g. [17], this notion is refined into two notions: receipt-freeness, which holds if a protocol does not require receipts to function, and uncoercibility which holds if receipts cannot be constructed. However, in this paper receipt-freeness means that voters cannot construct a proof of how they voted. These two concepts seem closely related, but as Chaum shows in [4], a voting protocol may give tickets (which seem like receipts) to voters without the voter being able to use these to prove how she voted. The reverse is also possible: a voting protocol need not supply voters with receipts in order for voters to be able to construct proofs of how they voted. An example of this is given by the attack on Benaloh and Tuinstra, mentioned in [11].

We note that this attack is not targeted at the core of BT, but exploits an auxiliary procedure in which voters prove that their encrypted vote is either a ‘0’ or a ‘1’. This auxiliary problem can be mitigated without breaking receipt-freeness, e.g. by applying the technique used in [2, Section 10.3.2]. We conclude that receipt-freeness of the core of BT remains unchallenged. Below we will

argue that the essence of the BT protocol indeed satisfies the receipt-freeness requirement.

The remainder of this paper is organised as follows: Section 2 describes related work. In Section 3, the notion of receipts in online voting systems is examined in depth. Section 4 formalises the notion of receipts, and expresses receipt-freeness in a process-algebraic setting. In Section 5, the application of this formalism to various protocols (BT, SK95, HS, ALBD and RIES) is discussed. Section 6 presents conclusions and possible directions for future research.

2 Related Work

In [8], an earlier formalisation of receipt-freeness is proposed. This formalisation does not define what constitutes a receipt, but instead relies on observational equivalence between a system with a colluding voter and a system without a colluding voter. A disadvantage of this formalism is that while it may be used to establish receipt-freeness of a protocol, it offers little aid to identify receipts when these are present. Our approach focuses on establishing what can constitute a receipt. This enables the identification of receipts, and provides a heuristic to take receipts into consideration in the early stages of designing a protocol.

Many voting protocols have been developed over the years. FOO [9] is well-known and has served as a base for various implementations (e.g. [7]). Its main goal is to allow voters to vote anonymously. RIES [12] has been used in two water management board elections in the Netherlands, handling over 70,000 votes in one case (making it one of the largest online elections). Where FOO and RIES are geared towards implementations, CFSY [5] and CGS [6] are aimed at providing desirable properties. The protocol described in CFSY provides information-theoretic privacy, universal verifiability and robustness. The main difference with CGS is that CGS allows a significant reduction in workload per voter, but offers only computational privacy.

Since Benaloh's and Tuinstra's work, various receipt-free voting protocols have been proposed, such as [18,1,11]. Other research has been aimed at providing receipt-freeness in more generic settings (as opposed to a single protocol), such as [16], which focuses on mix-nets, and [17], which proposes the use of an external trusted source of randomness.

The mechanisms to prove receipt-freeness given in these works are aimed at the specific topics covered. There seems to be little work towards a general formalisation of receipt-freeness that provides a uniform method to establish the absence or presence of receipts. It is the aim of the present paper to provide such a formalisation.

In the typical set-up adopted here, the adversary under consideration is an outside observer. Juels et al. mention several privacy-related attacks on voting protocols in [14]. Privacy and receipt-freeness are closely related, however, as stated in [14], receipt-freeness does not prevent these attacks. Juels et al. offer a stronger concept, which they call coercion-resistance, that offers receipt-freeness and prevents these attacks. Like [8], the work of Juels et al. enables establishing

coercion-resistance, but their computational adversary model provides no indication of what constitutes a violation of this property when it is successful.

This paper focuses on receipt-freeness (hence, it does not consider these privacy-related attacks) and on providing a constructive approach to determining receipt-freeness.

3 Receipts in Online Voting Schemes

Intuitively, a receipt r is an object that proves that a voter v cast a vote for candidate c . This means that a receipt r has the following properties:

- (R1) r can only have been generated by v .
- (R2) r proves that v chose candidate c .
- (R3) r proves that v cast her vote.

The difference between property (R2) and property (R3) is highlighted by the following example. Consider an election with paper ballots, where the ballots authenticate voters. In this setting, a voter could fill in a ballot, leave the voting booth with the ballot and show the ballot to anyone. This would satisfy properties (R1) and (R2), while clearly the voter did not cast a vote! Thus, a receipt must also prove that the voter cast her vote, which provides the justification for introducing (R3).

Although rather intuitive, the above properties do not catch receipt-freeness in non-1V1V settings. Consider, for example, the cheating in Italian elections for multiple posts (that are not 1V1V) as mentioned by Benaloh and Tuinstra in [3]. The authentication in that attack is provided by assigning each voter a permutation of posts. The voter authenticates herself by listing her choices in the assigned order on the ballot. Since anyone can construct such a permutation, this is usually not sufficient to satisfy (R1) because of the small number of permutations involved.

Receipts are regarded as undesirable objects in voting systems, for the following reason: if receipts can be created, an attacker can coerce (e.g. threaten) or entice (e.g. bribe) voters to vote in a manner of his choosing, and the voters can prove that they complied with this manner. This would mean that the result of an election no longer reflects the consensus of the voters, which is the primary goal of a voting system.

For many existing voting systems generation of receipts is possible. We illustrate this for the FOO protocol [9]. The main goal of the FOO protocol is to provide anonymity for the voters. This is achieved by the use of anonymous channels and blind signatures. Blind signatures are a cryptographic mechanism that allows a party to encrypt a message, have the encrypted message signed by another party and then remove the encryption, whilst preserving the signature.

The FOO protocol works as follows: First, a voter has her blinded, encrypted vote signed by a registration authority. Next, she removes the blinding and sends the signed, encrypted vote anonymously to the counter. After all votes have been received, the counter publicises an enumerated list of all received votes. The voter

then sends her decryption key (and associated list number) to the counter. After the counter has received all keys, he updates the public list to include the keys.

There is an obvious receipt here, given the public list: the key k used by the voter and its number ℓ in the list. This pair (ℓ, k) matches the requirements for receipts: k authenticates a specific voter (satisfying (R1)); using k , the ℓ th entry can be shown to be a vote for a specific candidate (satisfying (R2)); and since the pair (ℓ, k) is on the list, this refers to a cast vote (satisfying (R3)).

It should be noted that if the above receipt is shown to have been generated before the keys are published, it remains a valid proof after the publication of the keys. One way to prove that the receipt was constructed before publication is to use timestamping. As timestamping can be used in general, we adopt the view that the protocol is not receipt-free if at any time a voter can create a receipt.

4 Formalisation

In this section the notion of receipts will be formalised and subsequently used to specify what receipt-freeness means formally.

The architecture of 1V1V voting protocols is usually built from the following fundamentals:

- agents taken from the set \mathcal{A} , voters $Vot \subseteq \mathcal{A}$
- choices or candidates in Can , ballots in \mathcal{B} , and results (multisets of choices) in $\mathcal{M}(Can)$
- received ballots in \mathcal{RB} , from which the result will be computed
- a choice function $\Gamma: \mathit{Vot} \rightarrow Can$, which specifies how the voters vote
- a function $tally: \mathcal{P}(\mathcal{RB}) \rightarrow \mathcal{M}(Can)$, which gives the election result for a set of received ballots.

To denote receipts, the following notation is used:

- the set of receipts Rcpt
- $\mathit{Terms}(v)$, the set of all terms that a voter $v \in \mathit{Vot}$ can generate
- authentication terms $\mathit{AT}(v)$ for each voter, such that

$$t \in \mathit{AT}(v) \implies \forall w \neq v: t \notin \mathit{Terms}(w)$$

we put $\mathit{AT} = \{\mathit{AT}(v) \mid v \in \mathit{Vot}\}$

- a function $auth: \mathit{AT} \rightarrow \mathit{Vot}$, which returns the unique voter that created an authentication term.

A voting system is a system which takes a choice function as input and returns a set of received ballots as output.

Definition 1. *Given the above ingredients, a voting system \mathcal{VS} is a mapping: $\mathcal{VS}: (Vot \rightarrow Can) \rightarrow \mathcal{P}(\mathcal{RB})$.*

The result of \mathcal{VS} for choice function Γ is given by: $tally(\mathcal{VS}(\Gamma))$.

We have occasion to use the following auxiliary receipt decomposition functions:

- $\alpha: Rcpt \rightarrow \mathcal{AT}$, which extracts the authentication term from a receipt
- $\beta: Rcpt \rightarrow \mathcal{B}$, which extracts the ballot from a receipt
- $\gamma: Rcpt \rightarrow Can$, which extracts the candidate from a receipt.

Note that these functions depend on the structure of receipts, and thus vary from voting system to voting system. For example, for the FOO protocol discussed above, receipts are of the form (ℓ, k) if \mathcal{RB} was known. Assuming that elements of \mathcal{RB} are of the form $(num, \{vote\}_k)$, this means that in FOO (where $\mathcal{B} = \mathcal{RB}$):

- $\alpha(\mathcal{RB}, (\ell, k)) = k$,
- $\beta(\mathcal{RB}, (\ell, k)) = (\ell, e) \in \mathcal{RB}$, for some encrypted vote e , and
- $\gamma(\mathcal{RB}, (\ell, k)) = c$, such that $\{c\}_k = e$.

Using the above ingredients, the notions (R1) to (R3) can be expressed as follows.

Definition 2. *A receipt r for voter v concerning candidate c has the following properties:*

- (r1) $\alpha(r) \in \mathcal{AT}(v)$
- (r2) $\gamma(r) = \Gamma(v)$
- (r3) $\beta(r) \in \mathcal{RB}$.¹

Thus, a valid receipt r of voter v for candidate c can now be characterised as follows: $auth(\alpha(r)) = v \implies \gamma(r) = \Gamma(v)$ (satisfying (R1) and (R2)) and $\beta(r) \in \mathcal{RB}$ (satisfying (R3)).

Since $\gamma = \Gamma \circ auth \circ \alpha$ satisfies $auth(\alpha(r)) = v \implies \gamma(r) = \Gamma(v)$, we can now formulate when a voting system \mathcal{VS} allows receipts.

Definition 3. *A protocol $\mathcal{VS}(\Gamma)$ grants receipts in $Rcpt$ with derived functions α, β, γ iff*

$$\gamma = \Gamma \circ auth \circ \alpha \wedge \exists \varrho \in \text{Vot} \rightarrow Rcpt: \forall v \in \text{Vot}: \beta(\varrho(v)) \in \mathcal{RB}$$

As is obvious from the definition, γ can be seen as an abbreviation for $\Gamma \circ auth \circ \alpha$. Therefore, to determine presence of a receipt in a given protocol (for given $\Gamma, auth$), only α and β need to be defined.

The advantage of the above generic formalism is that it covers all receipts, also ones constructed by side-channels. The obvious disadvantage is that it does not provide us with a procedure to verify receipt-freeness of a given protocol. Note that a receipt is derived from a particular execution of a voting protocol, also referred to as a run. Only the public and private information exchanged during

¹ This assumes that $\mathcal{RB} \subseteq \mathcal{B}$. In case there are additional operations on the ballot after the voter sends her ballot (such that $\mathcal{RB} \not\subseteq \mathcal{B}$), which the voter cannot apply to her ballot herself, an auxiliary partial function $\phi: \mathcal{B} \rightarrow^* \mathcal{RB}$ is introduced; the requirement then is formulated as $\phi(\beta(r)) \in \mathcal{RB}$. Unless otherwise mentioned, $\mathcal{RB} \subseteq \mathcal{B}$.

the protocol run can be considered a building block of an associated receipt. Information not present in the run does not qualify for this.

In our approach to abstract voting protocols, all data is represented by terms. If we limit the notion of receipts to terms (i.e. $Rcpt \subseteq Terms$), procedural verification of receipt-freeness becomes possible by examining the suitability of terms as receipts. However, by limiting the formalism to terms, the ability to detect receipts constructed using side-channels (e.g. the interval between sending messages) is lost. In the remainder of this paper, we only consider receipts that are terms.

The exact syntax of terms depends on the specific protocol, however, in general terms it can be described by the following:

Definition 4. *Given*

- a set of choices or candidates Can
- a set of plaintexts PT (PT_i for specific agent i)
- a set of keys $Keys$ ($Keys_i$ for specific agent i)
- a set of functions $Func$ ($Func_i$ for specific agent i)

the class of terms of agent i is defined by

$$m_i ::= c \in Can \mid pt_i \in PT_i \mid k_i \in Keys_i \mid (m_1, m_2)_i \mid \{m\}_k^i \mid f(m)_i, f \in Func_i$$

which denotes respectively a candidate, a plaintext, a key, tupling of two terms, encryption of a term or function application to a term.

We write $t \in t'$ if t is a subterm of t' .

We can now formulate what extraction of a term means: For all functions F which extract one subterm from a term, it obviously holds that $F(t) \in t$. Therefore, we have the following observation.

Lemma 1. (*SUBTERMS*) For all terms $t \in Rcpt$: $\alpha(t) \in t \wedge \beta(t) \in t$.

The definition of $\mathcal{AT}(v)$ directly leads to

$$t \in t' \wedge t \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v)$$

which in turn leads to the following.

Corollary 1. It holds that $auth(\alpha(r)) = v$ iff $r \in \mathcal{AT}(v)$.

5 Application

The purpose of this section is to illustrate how to apply the formalism from the previous section to existing protocols. We provide a high-level analysis of the receipt-freeness of various voting protocols.

5.1 Application to Benaloh-Tuinstra

Since we are particularly interested in receipt-freeness, we restrict our examination of BT to just those parts related to receipt-freeness. As secret-sharing is not related to receipt-freeness, we focus on the ‘scaled-down election protocol’ as opposed to the protocol for multiple authorities. The proof of receipt-freeness of this specific protocol has not been disputed as far as we are aware. BT uses probabilistic encryption, which we denote by $x \in E(m)$ if x is an encryption of m .

The protocol distinguishes the following parties: voters $v \in \text{Vot}$; an authority A , who instantiated a probabilistic encryption scheme with parameter n , encryption function $E()$ and decryption function $D()$, and a beacon BC . The role of the beacon is to provide random bits, which are used in zero knowledge proofs (ZKP). Public communications between A and B are denoted by $s_{a \rightarrow b}$ (send) and $r_{a \rightarrow b}$ (receive), private communication (communication over a private, untappable channel) is denoted by $p_{a \rightarrow b}$. A superscript ‘*’ denotes communication of BT’s ZKP of the message instead of the actual message, so $p_{a \rightarrow b}^*(m)$ means A sends a ZKP of m over a private channel to B . As proven in [3], no terms of this ZKP can identify the voter’s ballot (i.e., for all such terms $t, \forall b \in \mathcal{B}: b \notin t$). Processes can be guarded: $b \rightarrow P$ denotes the process that can only execute if b is true.

The voting authority A takes the following steps per voter v : A sends V an encryption of ‘0’ and an encryption of ‘1’ in random order, and, over a private channel, a ZKP of which is which. The voter then casts her vote by returning the received value corresponding to her choice. This is modelled as follows (note that $\min(), \max()$ are used here to provide a random order of the two encryptions).

$$A(v) = \sum_{x \in E(0), y \in E(1)} s_{a \rightarrow v}(\min(x, y), \max(x, y)) \cdot p_{a \rightarrow v}^*(x \in E(0) \wedge y \in E(1)) \cdot (r_{v \rightarrow a}(x) + r_{v \rightarrow a}(y))$$

A voter v thus receives two encryptions over a public channel and a ZKP over a private channel proving which ciphertext denotes ‘0’. The voter then sends the encryption corresponding to the vote of her choice (given by $\Gamma(v)$) over a public channel. This is modelled by

$$V = \sum_{x, y} r_{a \rightarrow v}(x, y) \cdot \sum_{i \in \{0, 1\}} p_{a \rightarrow v}^*(x \in E(i) \wedge y \in E(1-i)) \cdot (\Gamma(v) = i \rightarrow s_{v \rightarrow a}(x) + \Gamma(v) = 1-i \rightarrow s_{v \rightarrow a}(y))$$

In examining the possible terms for voter v , note that v performs three actions: a receive over a public channel; a non-transferable receive over a private, untappable channel; and a send of a subterm of the message received in the first action over a public channel. It is clear that the first receive action does not provide an authentication term (communications over public channels can never constitute authentication terms). Hence, the last send action cannot communicate an authentication term. These actions thus cannot provide a term to satisfy the first conjunct of Definition 3 (since these terms are not in \mathcal{AT} , they are not in the domain of α).

The only possible terms which could supply a receipt are therefore the terms used in the ZKP. Since for all these terms t , we have that $\nexists b \in \mathcal{B}: b \in t$, the SUBTERMS lemma, Lemma 1, cannot be satisfied by any of these terms. Hence, none of the terms used in the protocol can act as a receipt.

5.2 Receipt-Free Voting Using Mixnets

The well-known SK95 protocol [18] uses a mixnet to provide anonymity. The protocol shuffles all possible votes and uses secure, untappable channels to the voter to prove that the mixnet functions correctly. Due to the untappability, there are no side-channel attacks on these proofs. Once again, these proofs are ment to provide the voter with the ability to prove different orderings of the shuffled vote. This is done using “chameleon blobs” – zero knowledge bit commitments that can be opened by a verifier in both ways. Because of this our analysis will use the assumption that these proofs cannot be exploited to provide receipts.

SK95 works as follows: The mixnet shuffles all possible votes and provides a proof of correctness of shuffling and a proof of the ordering of the votes to the voter, over secure, untappable channels. The voter submits the vote corresponding to her choice to the mixnet for the counter.

It is obvious from the above description that, aside from the proofs, voters possess no distinguishing knowledge. This means that no terms outside the proofs provide a means to distinguish voters, i.e. no term outside the proofs is in the set $\mathcal{AT}(v)$. Since we assumed that the proofs cannot be exploited to provide receipts, this lack of authenticating terms leaves us with no means to construct an appropriate function $\alpha: Rcpt \rightarrow \mathcal{AT}$, since $\mathcal{AT} = \emptyset$. Thus, the protocol is receipt-free.

Note that to fully prove receipt-freeness of SK95, obviously the transmitted proofs have to be considered as well. These proofs do provide authentication terms, but these terms are not linked to ballots, i.e.

$$\forall t \in Terms(v): t \notin \mathcal{AT}(v) \vee \nexists b \in \mathcal{B}: b \in t$$

which violates the SUBTERMS lemma.

5.3 Receipt-Free Voting Using Homomorphic Encryption

In the work by Hirt and Sako [11] the shuffling of SK95 is applied to protocols based on homomorphic encryption. In this respect, the HS protocol resembles the SK95 protocol. They propose a voting protocol which works as follows: each valid vote is encrypted in a deterministic way. Then, each authority takes the set of encrypted votes supplied by the previous authority, reencrypts each vote and outputs the votes in a random order. Again, each authority transmits a proof of correctness of this shuffle is transmitted publicly and the permutation is privately (untappably) communicated to the voter. The correctness of the permutation is privately proven to the voter using a designated-verifier proof.

Due to the conceptual similarities with SK95, we will not delve deeply into the analysis of HS here. We note that the analysis of SK95 applies to HS as well:

voters cannot be distinguished except by terms used in proving the shuffle, and assuming these proofs do not provide receipts, the protocol is receipt-free. Since the proofs are for a designated verifier, these proofs cannot be used to convince anyone but the voter of the order of the votes. Thus, again,

$$\forall t \in \text{Terms}(v): t \notin \mathcal{AT}(v) \vee \nexists b \in \mathcal{B}: b \in t$$

and thus the proposed protocol is receipt-free.

Note that Schoenmakers proposed an attack on this scheme, as mentioned in [14]: The coercer can force a voter to vote randomly. However, as mentioned in the introduction, this does not violate receipt-freeness and therefore it is not further considered.

5.4 Receipt-Freeness Despite Signing

The voting protocol ALBD, proposed by Aditya et al. in [1], differs from BT, HS and SK95 because it requires voters to sign votes. This means that there exist deliberate authentication terms for this protocol. On top of that, voter-supplied randomness is used to encrypt their vote in violation of the analysis in [17, Section 2.3]. However, receipt-freeness is attainable as these authenticating terms are only used in communications over an untappable channel. Assuming the administrator never discloses any knowledge of these terms, they cannot be linked to cast ballots.

We use the following notation in the process description: $E(m, k)$ denotes a random encryption of message m using key k , and $R(m, k')$ denotes re-encryption of m with key k' . $\{m\}_{SK(v)}$ denotes the signing of message m by agent v . Again, $p_{v \rightarrow a}(m)$ denotes communications over a private, untappable channel and $p_{a \rightarrow v}^*(m)$ denotes a communication over the private channel consisting of a designated-verifier proof of m .

ALBD uses a two-way untappable channel. Vote casting works as follows: A voter encrypts her vote ($m = E(\Gamma(v), k)$), signs it ($s = \{m\}_{SK(v)}$), and sends it over the untappable channel to the administrator. After the vote has closed, the administrator publishes re-encryptions ($r = R(s, k')$) of all received votes in random order. The administrator sends each voter a designated-verifier proof of the correctness of the re-encryption of their vote (denoted by $\exists k': r = R(E(\Gamma(v), k), k')$). This is modelled as follows:

$$\begin{aligned} V(v) &= \sum_k p_{v \rightarrow a}(\{E(\Gamma(v), k)\}_{SK(v)}) \cdot \\ &\quad \sum_r s_{a \rightarrow v}(r) \cdot p_{a \rightarrow v}^*(\exists k': r = R(E(\Gamma(v), k), k')) \\ A(v) &= \sum_y p_{v \rightarrow a}(\{y\}_{SK(v)}) \cdot \sum_{k'} s_{a \rightarrow v}(R(y, k')) \cdot \\ &\quad p_{a \rightarrow v}^*(\exists k: R(y, k') = R(y, k)). \end{aligned}$$

The designated-verifier proof only involves communication from the administrator to the voter. Because of this and that it is a *designated-verifier* proof it cannot be exploited to acquire a receipt.

Receipt-freeness of ALBD holds, because $\beta(r)$ cannot be constructed, as

$$\forall t \in \mathcal{AT}(v): \nexists b \in \mathcal{B}: b \in t$$

This means that there is no term that authenticates a voter *and* points to a ballot.

5.5 Application to RIES

The RIES protocol [12] deserves mentioning since it was used in two instances of the Dutch regional water management board elections. RIES, however, has trivial receipts. The purpose of its discussion here is as a more detailed example of how receipts are caught by the proposed formalism. The protocol uses a central administrator.

RIES works in three stages: pre-election, vote casting and post-election. In the pre-election stage, voters are registered, voter keys are handed out, the total number of voters is announced and for each voter, a ballot is constructed. The ballot is created using a keyed hash-function ($H(k, m)$ for key k and message m) and a keyless hash function ($G(m)$ for message m) and lists first the hashed voter-id ($G(H(k_v, election_id))$, for a given voter v) and then in the specified order a hash of each candidate ($G(H(k_v, c))$, for $c \in Can$). Both hash-functions are publicly known. The ballots are published as a list B . Note that, due to the imposed order, it is known to which candidate each item on B belongs. This is captured by the function can , defined as $can(G(H(a, b))) = b$.

A voter v casts a vote for candidate c by sending her voter-id $H(k_v, election_id)$ and her vote $H(k_v, c)$ via an anonymous, encrypted channel to the administrator. After the elections close, a list of all received ballots \mathcal{RB} is published. Tallying is done by, for each $r \in \mathcal{RB}$, counting one vote for $can(G(r))$.

Obviously, the voter-id together with the cast vote constitutes a receipt. However, due to the application of hash function G , $\mathcal{RB} \not\subseteq \mathcal{B}$. Since all parties know G , a voter can also hash his cast vote with G . Usually, it is assumed that the used hash-functions are collision-free and that all voter keys are different. In this case the hash of the cast vote would be sufficient to act as a receipt. Given a tuple of voter-id a and hashed cast vote $b = G(H(k_v, can))$, α and β can be defined as:

- $\alpha((a, b)) = a$ (or b)
- $\beta((a, G(H(k_v, can)))) = G(H(k_v, can))$.

Concluding In case of the RIES scheme an explicit construction of receipts invalidates the receipt-freeness of the protocol. For the other schemes, the absence of receipts was argued based on requirements on the receipt structure.

6 Conclusion and Future Work

The SUBTERMS lemma, as exploited above, establishes a method to prevent receipts: check that for all terms, the property of receipts does not hold. As seen in the analysis of voting protocols, this is done by assuring that all terms that can be used to authenticate a voter cannot be used to identify a ballot.

The reverse is also true: all terms which can be used to identify a ballot cannot authenticate a voter.

The analysis of BT, SK95, HS and ALBD shows that these protocols are receipt-free. The example of FOO and the analysis of RIES demonstrate that these protocols are not receipt-free. The receipt in RIES remains valid indefinitely. Although the keys used in FOO are revealed, this does not prevent construction of a receipt that remains valid by using timestamping.

A further line of research is to extend the intruder model to include the possibility of one or more authorities colluding with the intruder. This can be extended to the case where all authorities, but not the voter, cooperate with the intruder.

On a final note, the notion of receipt-freeness also has its applicability in online auctions, which we did not investigate for this paper. A formalisation of receipt-freeness for use in auction protocols is another possible line of future work.

Acknowledgements. The authors would like to thank Jos Baeten, Berry Schoenmakers, Wolter Pieters, Cas Cremers and Anna Zych for their insightful remarks on this subject and Simona Orzan for proofreading this paper.

References

1. R. Aditya, B. Lee, C. Boyd, and E. Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. In S. Katsikas, J. Lopez, and G. Pernul, editors, *TrustBus2004*, volume 3184 of *LNCS*, pages 152–161. Springer, 2004.
2. O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *PODC*, pages 274–283, Newport, Rhode Island, 2001.
3. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC'94*, pages 544–553. ACM, 1994. Montreal, Canada.
4. D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 02(1):38–47, 2004.
5. R. Cramer, M.K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In U.M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.
6. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In W.Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.
7. L.F. Cranor and R.K. Cytron. Sensus: A security-conscious electronic polling system for the internet. In *HICSS '97*, page 561. IEEE, 1997. Maui, Hawaii.
8. S. Delaune, S. Kremer, and M.D. Ryan. Receipt-freeness: Formal definition and fault attacks (extended abstract). In *Proceedings of the Workshop Frontiers in Electronic Elections (FEE 2005)*, Milan, Italy, September 2005.
9. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *ASIACRYPT'92*, volume 718 of *LNCS*, pages 244–251, 1992. Gold Coast, Australia.
10. D. Gritzalis. Principles and requirements for a secure e-voting system. *Computers & Security*, 21:539–556, 2002.

11. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In Bart Preneel, editor, *EUROCRYPT '00*, volume 1807 of *LNCS*, pages 539–556. Springer, May 2000.
12. E. Hubbers, B. Jacobs, and W. Pieters. RIES – internet voting in action. In R. Bilof, editor, *COMPSAC'05*, pages 417–424. IEEE, 2005.
13. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U.M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 143–154. Springer, 1996.
14. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In V. Atluri, S. De Capitani di Vimercati, and R. Dingledine, editors, *WPES'05*, pages 61–70. ACM, 2005.
15. C. Lambrinouidakis, D. Gritzalis, V. Tsoumas, M. Karyda, and S. Ikonomopoulos. Secure electronic voting: the current landscape. In D. Gritzalis, editor, *Secure Electronic Voting*, volume 7 of *Advances in Information Security*, pages 101–122. Kluwer, 2003.
16. B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In J.I. Lim and D.H. Lee, editors, *ICISC'03*, volume 2971 of *LNCS*, pages 245–258, 2003.
17. E. Magkos, M. Burmester, and V. Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In B. Schmid, K. Stanoevska-Slabeva, and V. Tschammer, editors, *I3E'01*, volume 202 of *IFIP conference proceedings*, pages 683–694. Kluwer, 2001.
18. K. Sako and J. Kilian. Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth. In L.C. Guillou and J.-J. Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.

Enhancing the Security and Efficiency of 3-D Secure

Mohammed Assora and Ayoub Shirvani

Anglia Ruskin University
Chelmsford, Essex CM1 1LL, UK
(m.assora, a.shirvani)@anglia.ac.uk

Abstract. Security is a major concern for all involved in E-Commerce and particularly in the case of online transactions using debit/credit card. Following the failure of Secure Electronic Transaction (SET), 3-D Secure is an emerging industry standard for online transaction security. Although 3-D Secure is a well designed protocol, it is still prone to some security problems and excessive numbers of messages which could reduce the speed of transaction. This paper uses a new cryptographic technique based on password only authentication and key exchange to present a new vision for 3-D Secure. The new vision covers the security problems and reduces the number of messages for 3-D Secure. Moreover, the new vision has the development ability to simulate SSL/TLS in its simplicity and at the same time abolishes SSL/TLS security glitches. This simplicity and security are the necessary factors for online transaction protocol to be the future standard.

Keywords: 3-D Secure; SSL/TLS; password based authentication and key exchange; online transaction security.

1 Introduction

Three-domain Secure or (3-D Secure) was developed by Visa [1,2]. The main purpose of the protocol is to provide authentication for the client by providing the ability for the issuer to authenticate the cardholder during online transaction. 3-D Secure is built on three domains; issuer domain, acquirer domain and interoperability domain. A brief description of these domains is as follows:

1. Issuer domain: this domain covers the relationship between the issuer and the cardholder. It consists of the issuer, cardholder and Access Control Server (ACS). The main function for ACS is to authenticate the cardholder during the transaction.
2. Acquirer domain: this domain covers the relationship between the merchant and acquirer. It consists of the acquirer and the merchant. The merchant must install a special program, known as merchant plug-in (MPI). The main function for MPI is to create and process the authentication messages.
3. Interoperability domain: this domain covers the relationship between issuer domain and acquirer domain. It consists of Visa Directory Server and Authentication History Server. The main function for Visa Directory Server is

to receive an enquiry from the merchant specific card number, forward the enquiry to ACS in issuer domain and then forward the answer from ACS to the merchant. ACS keeps a record in Authentication History Server for each attempt of authentication from ACS whether the attempt is successful or not.

1.1 3-D Secure

The 3-D Secure Scheme operates as follows [1]. The client browses the merchant's web site and decides to pay for goods or services. The protocol's steps are as follows, see figure 1:

1. The merchant's web server establishes a secure connection, by using SSL/TLS [12,13] channel, between the merchant and the client. The merchant asks the client to enter his/her bank details including his/her credit card details; these details are known as primary account number (PAN). Up till now, these steps are the common steps for normal transaction by using SSL/TLS. 3-D Secure continues as follows:
2. MPI queries Visa Directory Server if PAN is enrolled in 3-D Secure scheme and authentication is available. If PAN is not enrolled or no authentication is available, the process will stop and the transaction could only continue as a normal transaction, i.e. without 3-D Secure. To improve the performance, 3-D Secure implements the possibility of MPI's cache; in this cache MPI copies from Visa Directory Server the enrolled PAN numbers. In case of cache implementation, MPI stops 3-D Secure procedure directly if PAN is not registered. The following steps are for enrolled PAN in 3-D Secure scheme.
3. Visa Directory Server queries ACS for PAN if authentication is available.
4. ACS answers Visa Directory Server in addition to the uniform resource locator (URL) for ACS where MPI can connect directly to ACS.
5. Visa Directory Server forwards the answer to MPI.
6. MPI formats a payer authentication request (PAREq) which contains the transaction details. MPI signs this form by using his private key and posts it to ACS through the client web browser by using JavaScript.
7. MPI directs the client web browser, by using JavaScript, to ACS web site.
8. ACS asks the client to authenticate him/herself by using password or Visa smart card.
9. ACS signs the payer authentication result form (PAREs) by using his private key and posts it to MPI through the client web browser by using JavaScript. ACS sends a copy from the authentication result in addition to selected data from the transaction to authentication history server.
10. ACS redirects the client back to the merchant web site.
11. MPI Checks ACS certificate and then checks the PAREs signature.
12. If the authentication succeeds, the merchant completes the transaction in the normal way.

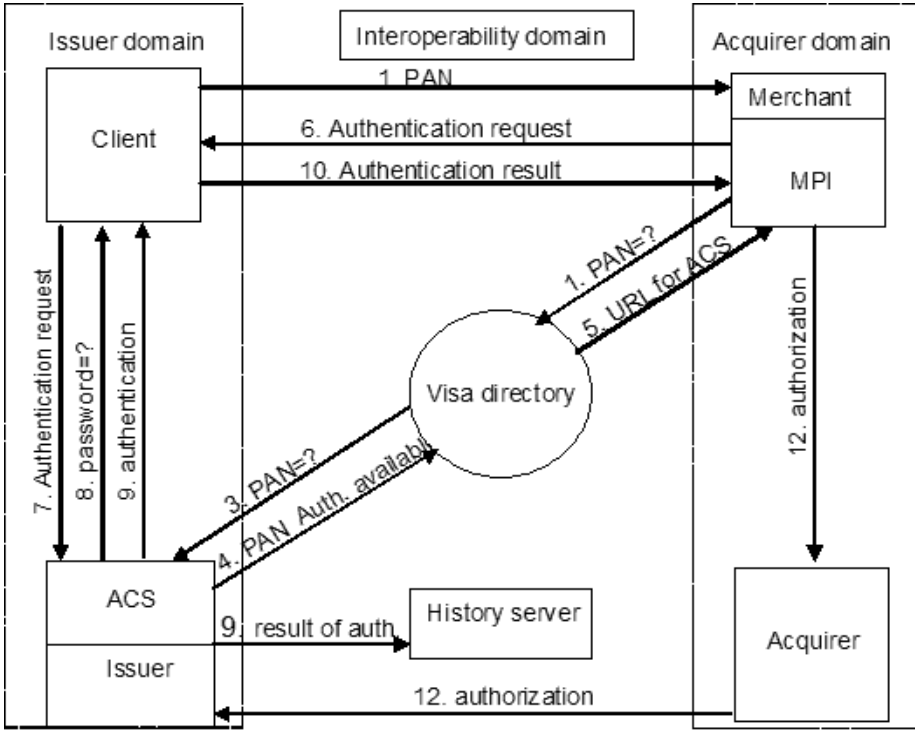


Fig. 1. 3-D Secure Transaction

2 Measuring 3-D Secure Against Online Transaction Security

The security requirements for online transaction include authentication, data confidentiality, data integrity, non-repudiation and freshness of the transaction. We will examine how much 3-D Secure fulfils these security requirements:

- Authentication for the entity: every party must be able to authenticate the entity with whom they are dealing. In normal transactions, by using SSL/TLS, there are only two parties, the client and the merchant, who have to authenticate each other. In 3-D Secure, in addition to the client and the merchant there are Visa Directory Server and ACS. 3-D Secure uses public key certificate for the merchant, Visa Directory and ACS for their authentication. The cardholder uses password or Visa smart card to authenticate him/herself to ACS. As a result, the authentication for entire parties is provided.
- Data confidentiality during and after the transaction: the data confidentiality is provided during the transaction because 3-D Secure uses a secure channel by using SSL/TLS. The confidentiality of user data after the

transaction depends on the confidentiality of the merchant database, which has the transaction details, but it is better than using naive SSL/TLS because the authentication secret value for the client is not part of this database. If 3-D Secure is widely used and the transaction is not possible without client authentication, then an attacker, who has access to this database, is not able to complete a transaction on behalf of the cardholder.

- Data integrity: 3-D Secure uses SSL/TLS channel to exchange transaction data. SSL/TLS provides data integrity for the exchanged data only during the transaction. After the transaction the integrity of the transaction data depends on the integrity of the merchant's database which contains the transaction data but it is better than naive SSL/TLS because the selected data from the transaction, which ACS keeps in the authentication history server, can be considered as means of backup for the parties in case of dispute.
- Non-repudiation: the authentication history server keeps a selected data from the transaction, from the merchant, in addition to the cardholder authentication result, so non-repudiation is provided.
- Freshness of the transaction: freshness of the transaction means after the transaction has been conducted, neither the merchant nor third party is able to repeat the same steps to perform other unauthorized transaction built on authorized transactions. If 3-D Secure is widely used and the transaction is not possible without client authentication, then the merchant or an attacker, who knows the client account details, is not able to complete a transaction on behalf of the cardholder.

As a result, 3-D Secure provides the security requirements for online transaction. If the authentication is conducted by using a password, there is no need for any additional software or hardware from the cardholder, no need for private/public key pair for the cardholder and all the connections are secure by using SSL/TLS.

3 3-D Secure Under Fire

From the first reading, we can say the problem of online transaction has been solved because, as mentioned above, 3-D Secure can provide all the security requirements and it is not expected to have a major implementation problem [3]. However, in the cardholder authentication process, step 8 in figure 1 above, the authentication for the far end of the connection, who is supposed to be ACS, is achieved by its public key certificate during SSL/TLS connection establishment. A faked merchant can carry out an attack against the client and reveal his/her secret authentication value by using one of these methods:

3.1 Man in the Middle Attack

After MPI gets the URL of ACS (step 5 in figure 1 above), MPI does not redirect the client web browser to ACS. Instead, MPI, or any web site under the faked merchant control, acts as a proxy for the client and the connection to ACS will go through MPI. In this case, MPI requests the page from ACS and forwards it to

the client and vice versa with the possibility to read and modify all the exchanged pages. If there is a secure channel, by using SSL/TLS, MPI makes two secure connections; one with ACS and the other with the client. The client can discover the attack by checking the origin of the page because it will give the name of MPI not ACS. Theoretically, the client can discover the origin of the page from the page's URL, the status line for the page and from the source of the page. Practically, the normal user, i.e. the majority of users, does not check these values. The check in 3-D Secure is more difficult because during the transaction the client is dealing with the merchant. Therefore, the merchant's URL is the correct URL and the URL is changed only during the authentication process, i.e. during entering the password. Finally and this is the worst, what should appear in the page's URL and the status line for the page can be changed to any desired value by using JavaScript. The common recommendation for this attack is to disable JavaScript in the web browser [4,5] but this solution is not possible in 3-D Secure because MPI and ACS use JavaScript to redirect the client web browser (step 7 and 10 in figure 1 above). Fortunately, the source of the page cannot be changed by using JavaScript but the source of the page is written in HTML. So it is not possible for non-programmer client to read this source and discover the real origin of the page. The ideal method to inspect this type of attack, in the case of SSL/TLS secure channel, is to inspect the certificate of the web site [5]. The certificate will clearly identify the owner of the certificate, i.e. who establishes the secure connection. Theoretically, checking the certificate is perfect solution but practically the majority of clients do not know what the public key certificate is or how to check the certificate. Furthermore, in 3-D Secure the client has three secure connections (see figure 1 above), first client merchant (step 1), second client ACS (step 8) and third client merchant (step 10). Consequently, this check should be performed three times. By using this attack, the faked merchant can reveal all the client account details including his/her authentication value. Moreover, because the merchant can modify the exchanged web pages between the client and ACS, the merchant is able to send to ACS and the client different copies of transaction details. Practically, the latter attack is not so desirable because it could put the merchant under suspicion.

3.2 False Redirection

In this scenario, MPI, for a faked merchant, instead of redirecting the client to ACS, step 8 in figure 7, redirects the client to a faked ACS (FACS) under his/her control. FACS is able to form a web page exactly the same as the genuine page because the merchant knows the transaction details and the client account details. 3-D Secure uses personal assurance message (PAM) [1] to reduce spoofing and a simple means of mutual authentication. PAM could not be considered as a strong method to authenticate ACS to the client because, firstly, PAM appears in clear text on the client's screen during the authentication so anyone can read PAM and, secondly, a faked merchant can use the client account details, from previous transaction and a faked client under his/her control to simulate the real client to obtain PAM from ACS. This attack can be conducted as follows:

1. The faked client uses the real client's account details to do a faked transaction with any merchant. The reason for choosing another merchant is to remove any suspicion of the faked merchant during the further attack. As previously mentioned, ACS keeps a record of all the authentication attempts.
2. The merchant redirects the faked client to ACS.
3. ACS provides PAM, in clear text, to the faked client and asks him/her to authenticate themselves
4. The faked client enters any password
5. The transaction is not completed because the password is not correct but PAM has been revealed.

In any further transaction from the client, the merchant can redirect the client to any faked web site to reveal his/her secret authentication value. To prevent this attack, the client should deal with it as man in the middle attack mentioned above by effectively checking the web site of ACS and the certificate of ACS when there is secure connection between the client and ACS.

4 Password-Based Authentication

The main goal for 3-D Secure is to authenticate the client. 3-D Secure presents many methods such as password, mobile phone and smart card. From these authentication methods, using only password has many advantages, such as being simple and easy to use, no additional hardware is required, convenience and traditional method for authentication. However, using a human-memorable password only authentication in the traditional method, which takes a hash function of the password, is insecure because of the low entropy spaces for the password that allow malicious guess attack and particularly the dictionary attack. Bellare and Merritt [6] first presented authenticated key exchange based on password protocol known as Diffie-Hellman Encrypted key exchange, DH-EKE. This protocol was built on a combination of public key and secret key cryptography. The main idea of the protocol is to deny the attacker from knowing if his/her guess of the password is correct or not. Since the presentation of the first protocol, many other protocols have been presented [7] which develop new ideas that make the protocol stronger and add some advantages to the original protocol. Password-based protocols for authentication and key exchange hold promise for the future and recently have received significant attention. Currently, there is draft for an IEEE standard [8] for some of these protocols in addition to RFC draft to build a TLS secure channel by using only password [9]. The integration process for password only authentication and key exchange in network applications such as FTP, telnet and SSL/TLS is active in many research groups and there are beta versions of the programs [17,18].

5 The New Approaches

Two approaches are presented in this section. The first approach tries only to address the security loopholes mentioned above. The second approach presents a new direction for 3-D Secure.

5.1 First Approach

This approach covers the security loopholes mentioned above and no change is required in 3-D Secure. During the authentication process, the connection between the client and ACS is protected in the actual protocol by using standard SSL/TLS, which is based on the server public key. If this connection is built on the new model of SSL/TLS, which uses cipher suite based on password from the client and ACS [9], the security loopholes, mentioned above, will be covered. From [14,15,16], it can easily be seen that man in the middle attack is impossible and false redirection does not leak any extra information about the password because ACS receives the same messages as any other adversary. In this paper, only two attacks are presented; [16] and many of similar protocols are proven secure against attack from passive and active adversary. The mobility of the client is still maintained, i.e. no need for any extra software or hardware. The only requirement is to implement any protocol using only password as means for SSL/TLS connection, as mentioned in [9]. In the author's opinion it is only a matter of when rather than if it will be done. Finally, it is important to mention that no conflict between using password method for SSL/TLS connection and other types of authentication, such as smart card, mobile phone, or any future type of authentication, in contrast, using password can support the authentication process.

5.2 Second Approach

Client redirection seems to be attractive, particularly for the client, because it simulates the traditional method of online transaction, by using naive SSL/TLS. Client redirection has a major advantage that all steps are automated for the client and no need for any extra hardware or software. However, in addition to the possibility of potential attack as mentioned above, there is the possibility of implementation difficulty and particularly the speed of transaction, since the number of messages is big and consequently the number of authentications is big with the possibility of many bottlenecks. This section tries to make an improvement for 3-D Secure to reduce the number of messages in addition to make it straightforward i.e. every party makes one connection with one party. Abdalla et al [10] present a three-party password-based protocol to authenticate the client through distrust server. In the following, a new protocol for online transaction security is built on the protocol in [10] and 3-D Secure; this protocol from now on will be called enhanced 3-D Secure, see Figure 2.

These abbreviations are used:

Z_q : is a prime-ordered group, where Diffie-Hellman problem is hard.

q : is order of group Z_q .

g : is a generato for Z_q .

φ, h, \hat{h} : are one way hash functions.

All the multiplications, additions and exponentiations are performed modulo q

1. The client browses the merchant web site and when he/she decides to pay for goods or services, a secure connection, by using SSL/TLS based on the merchant public key, is established. The client is asked to enter his/her account

- details including PAN. MPI checks with Visa Directory Server if authentication for PAN is available with the possibility of using MPI's cache [1].
2. If authentication is available for PAN, MPI sends 'change cipher spec' message [12,13] for SSL/TLS connection with the client and new cipher suite, which uses password-based protocol, is activated.
 3. The client's browser, CB, asks the client for his/her user name, c and password, pw , then CB computes $PW = \varphi(c, G, pw)$, where G is the merchant identifier.
 4. CB chooses a random number $x \in Z_q$, computes $X = g^x$, computes $X^* = X \times PW$ and sends X^* to MPI
 5. MPI chooses a random number $y \in Z_q$ and computes $Y = g^y$
 6. MPI establishes SSL/TLS connection with Visa Directory Server based on the public key of the two parties, i.e. client and server must have public key certificate.
 7. MPI sends c, PAN, X^*, Y and some selected data from the transaction to Visa Directory Server.
 8. Visa Directory Server connects with ACS through a special financial connection and passes the values c, PAN, X^*, Y and the selected data from the transaction.
 9. ACS calculates $X = X^*/PW$, chooses a random number $s \in Z_q$, computes $\alpha = X^s, \beta = Y^s$ and sends the values α, β to MPI through Visa Directory Server
 10. MPI computes $k = \alpha^y$, $authG = h(c, G, X^*, \beta, k)$ and sends β and $authG$ to CB, to continue SSL/TLS connection.
 11. CB computes $k = \beta^x$, $authG' = h(c, G, X^*, \beta, k)$ and accepts the result if and only if $authG = authG'$
 12. CB and MPI compute the session key for SSL/TLS session as $sk = h(c, G, X^*, \beta, k)$
 13. The merchant sends the transaction details, the client confirms the transaction and then the transaction is completed.

The merchant keeps the transcript (c, G, X^*, β) as proof of authentication from the client and authorization for the transaction from ACS.

Let us first compare the efficiency of the new protocol with 3-D Secure. We will compare 3-D Secure steps, recall section 1.1 and figure 1 above, with the new protocol. Before the comparison, an important point should be clarified. The role of MPI cache is a little bit different between the two protocols. As there will be a direct connection between MPI and the client to ACS, so the steps 2-5 in figure 1 above are mandatory. Particularly, MPI should know the URL where to redirect the client for authentication. If MPI uses cache, this step can be avoided only if PAN is not enrolled and 3-D Secure will not be used [1]. In the new protocol steps 3 and 4 in figure 2 are necessary only if no cache is used. Any required information to complete the transaction such as PAN, the merchant identification and authentication such as Acquirer BIN, Merchant ID and Merchant Password, which are included in Verification request (VEReq) form, can be sent later with the message 5 in figure 2. In case of error, such as

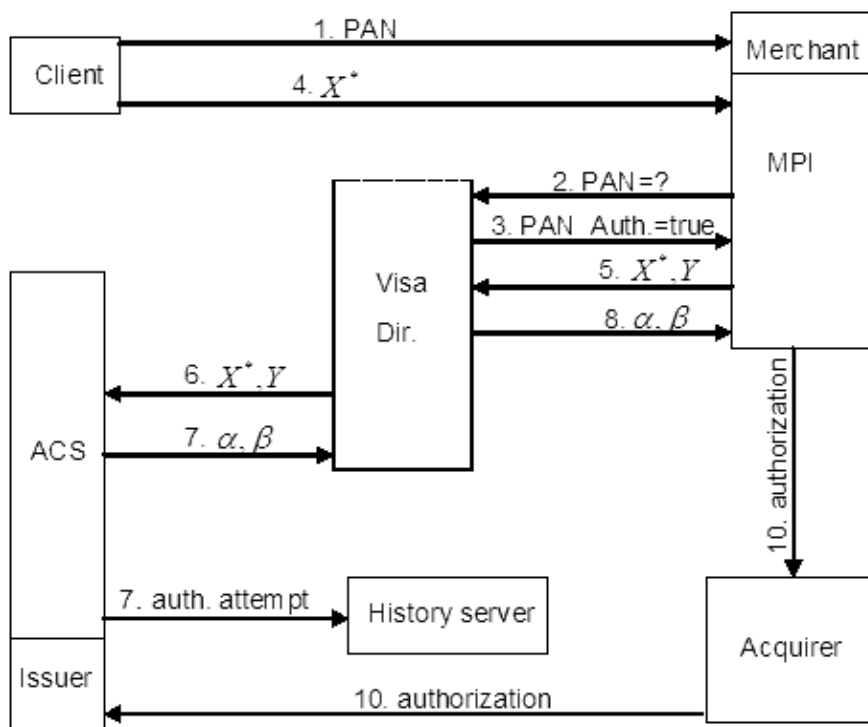


Fig. 2. Enhanced 3-D Secure Transaction

no authentication being available from ACS even when PAN is enrolled, which could happen in 3-D Secure and in the new protocol, MPI in the new protocol simply returns to cipher suite which is built on the server public key. Using the cache in the new protocol is highly recommended.

In the comparison underneath, we consider that the MPI cache is implemented as recommended in [1] and PAN is enrolled in 3-D Secure scheme:

Step 1: The merchant establishes SSL/TLS channel with the client and the client enters his/her account details (PAN). This step is nearly the same in the two protocols. Because MPI knows that PAN is enrolled in 3-D Secure scheme, from its cache, MPI, in the new protocol, can directly change SSL/TLS connection to a cipher suite that is built on password-based protocol. The client is asked to enter his/her password, X^* is calculated and sent to MPI. From the client view there is only the password that is extra to 3-D Secure. Entering the password is done directly after the client submits PAN information. From the MPI view there is only one extra check for PAN in its cache. ACS and Visa Directory are not involved in the transaction yet. As a result the time of this step is nearly the same in the two protocols.

Steps 2: In this step, MPI in 3-D Secure queries for PAN if it is enrolled in 3-D Secure scheme. In addition to that, MPI will send VEReq form, which contains the merchant identification and authentication. In the new protocol, MPI sends the same information in addition to X^*, Y and some selected data from the transaction.

Step 3: In this step, Visa Directory queries ACS if PAN is enrolled in 3-D Secure scheme, i.e. forwards the coming message from MPI to ACS. This step is the same in the two protocols.

Step 4: ACS answers Visa Directory. This step is nearly the same in the two protocols. In the new protocol ACS calculates the values α, β , which is simple mathematical operations and will not cause any difference between the two protocols.

Step 5: Visa Directory forwards the previous message (step 4) to MPI. This step is exactly the same in the two protocols.

Steps 6-11: The important contribution in this investigation is the elimination of steps 6-11 which contain the direct connections between MPI and the client to ACS mentioned in section 1.1. These steps, which form the majority of time and resource consuming for 3-D Secure operations, do not exist in the new protocol.

Step 12: Authorization from the acquirer. This step is the same in the two protocols.

The theoretical calculation estimates the number of steps in the new protocol will be reduced by up to half the steps in 3-D Secure. Speed of transaction will be improved by reducing the number of messages. The percentage of this improvement can be determined only by practical implementation of the protocol which is a good idea for further work in this project.

It can be observed that the proposed protocol is able to achieve all the goals of 3-D Secure. The major advantages and features for the protocol can be summarized as follows:

- No change to the theoretical basis for the protocol in [10], so the proof of security in [10] is still valid. Particularly, as the protocol does not leak any information about the client's password even for the merchant and Visa Directory server.
- As 3-D Secure, no need for any software or hardware for the client, to authenticate him/herself by using password, the web browser should support cipher suit for SSL/TLS connection based on password only protocol such as [9]. Although, [10] and the protocol above are built on one mask Diffie-Hellman key exchange protocol [11], which is password based protocol for two parties it is not difficult to adopt the protocol for any other protocol such as mentioned in [8].
- No conflict with other types of authentication. Broadly, if the authentication is built on shared secret cryptography, this shared secret can be considered as the password in the protocol above and no need for any change. If the authentication is built on public key cryptography, it will not be a problem if

the authentication value is passed through the merchant and Visa directory server.

- MPI software for the new protocol is simpler than MPI software for actual 3-D Secure. The authorization process is transparent for MPI, i.e. MPI deals with one server and does not require knowing who ACS is. A worthwhile idea is to search if the acquirer can perform the role of Visa Directory for its registered merchants.
- The protocol is transparent to the client. He/she uses the same steps, which is used in naive SSL/TLS, for online transactions. He/she only authenticates him/herself by a password reasonably secure. It should only be secure against online guess attack.
- ACS in the new protocol is background server. It only has connection with Visa Directory Server and this gives many advantages such as being more secure, faster, and with no need for many authentications process. Also there is the possibility of an integrated system which consists of servers that have access to a huge database of clients' authentication values.
- In addition to the simplicity of the protocol, the redundancy is possible for the servers, Visa Directory and ACS, to avoid any possibility of bottleneck.
- A disadvantage for the protocol is that because the authentication is performed in an implicit way, the protocol should be repeated completely in case of error, i.e. if the client makes a mistake in entering his/her password, the protocol should be performed again completely. A policy can be applied in the web browser to ask the client to enter the password and then to confirm the password. The web browser does not continue if they are not identical. The worse, if the client forgets his/her password, no way to add hint or recovery for the password; the client should use the enrolment procedure [1] to recover his/her password.
- The protocol uses implicit authentication, i.e. ACS does not know if the authentication process is successful or not, so online guessing attack for the password is easier to be performed. ACS, by using authentication history server, should perform an audit for suspicious transactions; for example, too many authentication requests for a client and, particularly, incomplete transactions.

6 Conclusion

3-D Secure is good protocol for online transaction. It tries to combine secure electronic transaction (SET) with SSL/TLS to improve the security of naive SSL/TLS as a means of online transaction security and at the same time do not stuck in the implementation issues as SET. 3-D Secure reasonably achieves its main goals. However, there are still some security risks such as threats of man in the middle attack and false redirection for the client during the authentication. Moreover, the number of messages is big and this could reduce the speed of transaction in addition to the possibility of bottlenecks. By using a new technique of cryptography, authentication and key exchange based on memorable password

and implementation of this technique in SSL/TLS, a remedy for the security glitches has been achieved as the first step. A new vision for 3-D Secure has been designed that makes it more secure and simple. The theoretical bases for the new approaches are ready. The practical implementation does not require any additional hardware over actual 3-D Secure. In fact, the new approaches use the hardware more efficiently. Software implementation for the new approaches starts by implementing password only authentication and key exchange as cipher suit for SSL/TLS. This step has been already started [9,17,18] but it could need some time for adaptation by commercial web browser client providers. A worthwhile idea for further research is to investigate if the acquirer is able to serve as Visa Directory Server in the new approach of 3-D Secure system. This scheme has many advantages such as enhanced security, minimal software alterations and no hardware changes required to the actual setting which uses naive SSL/TLS for online transaction security. A possible disadvantage of this scheme is that the acquirer could be a potential bottleneck for the system.

References

- [1] Visa International Service Association. 3-D Secure Protocol Specification: Core functions version 1.0.2, July 2002.
- [2] Visa International Service Association. 3-D Secure Protocol Specification: System over-view version 1.0.2, July 2002.
- [3] Jarupunphol, P. and Mitchell, C. J. Measuring 3-D Secure and 3-D SET against ecommerce end-user requirements. In Proceedings of the 8th Collaborative electronic commerce technology and research conference. National University of Ireland, Galway, June 2003, pp.51-64.
- [4] Aviel D. Rubin, Daniel Geer, and Marcus J. Ranum. Web security sourcebook A complete guide to web security threats and solutions. John Wiley and sons. 1997
- [5] Anup k. Ghosh. E-commerce security: weak links, best defenses. John Wiley and sons. 1998
- [6] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In 1992 IEEE Symposium on Security and Privacy, pages 72-84, Oakland, CA, May 1992. IEEE Computer Society Press.
- [7] Colin Boyd, and Anish Mathuria. Protocols for authentication and key establishment. Springer-Verlag. Berlin, Germany. 2003.
- [8] IEEE Standard 1363.2 Study Group. Password-Based Public-Key Cryptography. Available from: <http://grouper.ieee.org/groups/1363/passwdPK>.
- [9] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin. Using SRP for TLS Authentication. Internet draft-ietf-tls-srp-10. 06/10/05, expires 09/04/06
- [10] Michel Abdalla, Olivier Chevassut, Pierre-Alain Fouque, and David Pointcheval. A Simple Threshold Authenticated Key Exchange from Short Secrets. Advances in Cryptology - ASIACRYPT 2005, Volume 3788 of Lecture Notes in Computer Science, pages 566-584, Chennai, India, Dec. 4-8, 2005. Springer-Verlag, Berlin, Germany.
- [11] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. New security results on encrypted key exchange. 7th International Workshop on Theory and Practice in Public Key Cryptography, volume 2947 of Lecture Notes in Computer Science, pages 145-158, Singapore, March 1-4, 2004. Springer-Verlag, Berlin, Germany.

- [12] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0. Netscape. 1996.
- [13] RFC 2246. T. Dierks and C. Allen. The TLS protocol. version 1.0. IETF, January 1999.
- [14] T. Wu, SRP-6: Improvements and Refinements to the Secure Remote Password Protocol, Submission to the IEEE P1363 Working Group, Oct 2002.
- [15] RFC 2945. T. Wu. The SRP Authentication and Key Exchange System. IETF, September 2000.
- [16] T.Wu. The secure remote password protocol. In proceeding of the internet Society Net-work and Distributed System Symposium, Pages 97-111, March 1998.
- [17] The Stanford SRP Authentication Project, Stanford University,USA <http://srp.stanford.edu/>
- [18] Open SSL project. <http://www.openssl.org/>

Designing and Verifying Core Protocols for Location Privacy

David von Oheimb and Jorge Cuellar

Siemens Corporate Technology, Munich, Germany

Abstract. Geographic privacy services provide location information on roaming targets to location recipients via location servers, in a way that protects the privacy of the individuals involved. In this paper we propose and discuss new protocols representing the core of Geopriv, with particular focus on the security requirements stated in the IETF's RFC 3693. Using the AVISPA tool, we check that these requirements, namely anonymity against the location server, as well as confidentiality, integrity, and authenticity of the location information, are actually met. In the design phase of such protocols, numerous variants are to be considered and evaluated. Here the use of model checkers turns out to be very helpful in exploring the security implications quickly and precisely.

Keywords: Geopriv, location information, privacy, model checking, exploration.

1 Introduction

With the widespread use of mobile devices like mobile phones and GPS receivers, *Location Based Services* offer convenient and commercially attractive ways to solve issues like “Please direct me to the nearest shopping mall.”, “Which time zone is my boss currently traveling in?”, or “Has the kid reached home safely?” However, location information needs to be gathered and transferred securely, protecting the privacy of the individuals involved.

This paper explores some of the basic protocols that can be used to transfer location data, respecting the authorization, integrity and privacy requirements. See the RFCs 3693 [CMM⁺04] and 3694 [DMMP04] for more background on requirements and threat analysis.

The IETF working group Geopriv [GWG06] has focused itself on

- the format of the *Location Information* to be sent (“*Location Object*”),
- the format of the *Privacy Rules* describing policies to be applied,
- particular cases of so-called “using” protocols, that is, protocols that carry Location Information about a mobile user (the “*Target*”) from a *Location Server* to a *Location Recipient*.

Nevertheless, to understand the requirements and goals of Geopriv, one needs to consider also protocols that are out of scope at the IETF. The pieces missing are protocols used to

- agree on pseudonyms and/or passwords for the Target and the Location Recipient, which are to be used by the policies,
- request, using the credentials just mentioned, the Location Information from a Location Server,
- transfer credentials to a Location Server,
- transfer the Privacy Rules to a Location Server,
- transfer the Location Information to an initial Location Server.

The main goals of this paper are to model a reference implementation of the overall system and to evaluate its security in a systematic and holistic way. This can be done only by including the just mentioned components.

The work described here has been done in an industrial context, at the intersection of standardization bodies and commercial implementors. The main principles of our design are:

- The most important requirements are to guarantee the correctness (integrity) and confidentiality of the Location Information. This requires authenticating the main entities of the protocol and securing the exchanged messages.
- A central role is played by user-controlled policies, which describe the permissions (or consent) given by the Target. The policies specify not only the time and place when Location Information may be released to whom, but also which component (or derived measure) of the information is to be released and in which granularity or accuracy.
- Whenever possible, the Location Information should not be linked to the identity of the user. Rather, the user is able to specify which local identifier, pseudonym, private identifier, or token is to be used instead. Although complete anonymity may not be appropriate because of legal constraints or because some location services do in fact need the explicit identification of the user, we argue that in most cases the location services may only need some type of authorization information and/or perhaps an anonymous identifier of the users, that may change as often as needed.
- To ease comprehensibility and implementation, our reference model should be as concise as possible. It shall clarify any issues left open by the RFCs. Particular solutions and alternatives shall be motivated and explained.

Note that the main challenge and novelty aspect here is the anonymity goal (identity protection), which inherently opposes the authentication requirements and makes the use of existing protocols with standard certificates etc. impossible.

Given the recent advances in protocol analysis by the project AVISPA [AH-03] and the availability of their tools, we have done the modeling in the High Level Protocol Specification Language HPSL [CCC⁺04] and have conducted our analysis with the tools contained in the AVISPA package [AT-05].

For reasons of space and to avoid confusion by two different syntactic levels, we have decided to use for our presentation a pseudo-mathematical “Alice-Bob”-style notation that seems to be widely accepted or at least easily understandable without much further explanation. For more information on specifications in HPSL and on how to check them with the AVISPA Tool, please refer to the HPSL Tutorial [AVI05a] and the AVISPA User Manual [AVI05b].

2 General Design

As already mentioned, typical Geopriv protocols involve a *Target (T)*, whose *Location Information (LI)* is to be conveyed to a *Location Recipient (LR)* by a *Location Server (LS)*. A *Privacy Rule (PR)* defines the policy: who is allowed to learn whose location, under which circumstances, with which *Granularity (GR)*. The Granularity may be, for instance, the complete street address, the GPS coordinates up to a given precision, or just the time zone.

Although it is assumed that Location Servers adhere to the protocol and cannot be compromised by an attacker, at least some types of Location Servers are considered untrusted in the sense that they should not learn the real identities of the Location Recipient and of the Target. So when communicating with the Location Server, these parties do not use their real names but just pseudonyms and/or passwords. Each party can authenticate itself to the Location Server with the help of such a password or a signature related to the respective pseudonym. The pseudonyms and passwords, if any, form part of the Privacy Rules. As opposed to passwords, pseudonyms in general need not be kept secret.

It is assumed that Location Recipients do not abuse the Location Information they obtain, e.g. by publishing it. Yet in our analysis, we will consider not only standard sessions with honest Targets and Location Recipients, but also problematic sessions where the intruder is allowed to assume the role of either of these two. This means that his initial knowledge is augmented with the private keys that a legitimate player of the role usually has, enabling him to play that role properly. Such sessions themselves do not make much sense because the intruder is made a legitimate source or receiver of the Location Information. It is interesting, however, to see if such degenerate sessions can have a bad effect on standard sessions.

Usually, the Target subsumes the role of *Rule Maker*. In our presentation we further assume that the Target is also the *Location Generator*.¹ We also assume that the Location Server is the *Rule Holder*.

The exchange between the Geopriv entities can be divided into the following phases (or, sub-protocols).

Agreement. The Location Recipient and Target, who usually know and trust each other, exchange credentials like pseudonyms and passwords. Typically, this includes mutual authentication, as well as authorization of the Location Recipient.

Policy. The Privacy Rule is transferred from the Target to the Location Server, potentially via a separate Rule Holder. The main issue here is authorization of the Target and authentication and integrity of the Privacy Rule.

Location. The Location Server learns the current location of the Target, potentially via a separate Location Generator. The main issue here is secrecy, authentication, and integrity of the Location Information.

¹ A simpler — but in many cases not possible — alternative is that the Location Server senses the presence of the Target directly (still without knowing T's identity).

Information. The Location Recipient requests the location of the Target and receives an answer from the Location Server. The main issues here are authentication and authorization of the Location Recipient, as well as the secrecy, authentication, and integrity of the Location Information, this time from the perspective of the Location Recipient.

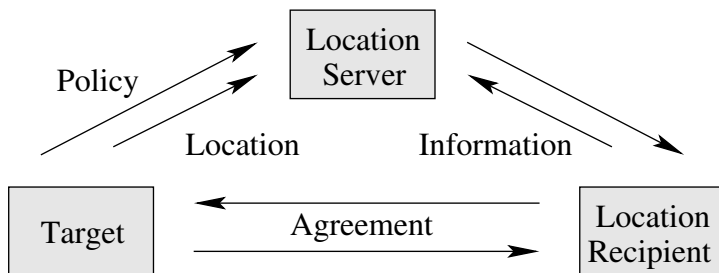


Fig. 1. Geopriv Structure

The general Geopriv structure is depicted in Fig. 1. The order of the first three of the four sub-protocols may vary, whereas — without loss of generality — we assume the order just given.

For each of the four phases, there are various ways to implement them. In this paper we describe in detail two Geopriv protocol variants that exhibit four differences distributed over all these phases. These differences are orthogonal to each other, such that in effect we implicitly cover $2^4 = 16$ variants.

3 Variant with Two Self-signatures

In our first variant, both the Location Recipient and the Target use a pseudonym and a self-signature. A *self-signature* is a digital signature where the sender’s public key (or a hash of it) is included in the part signed with the corresponding private key. No certificate is used to link the sender’s identity with the public key, but still the receiver can check that the sender holds the corresponding private key and therefore should be the owner of the public key. Self-signatures are used here to protect the anonymity of the Location Recipient and the Target when communicating with the Location Server. To this end, each of the two parties $X \in \{LR, T\}$ creates a public/private key pair $(P_X, P_X^{-1})^2$ and uses the hash of the public key $h(P_X)$ as its pseudonym Ψ_X . We use the letter ‘P’ rather

² We use the handy notion X^{-1} to refer to the private key related to the public key X , which does not imply that the former can be derived from the latter, but alludes to the fact that encryption using X can be inverted by decryption using X^{-1} , and signatures (by decryption) using X^{-1} can be checked (by encryption) using X .

than the usual ‘ K ’ for the these keys to emphasize that they are related to a pseudonym. The purpose of hashing is just to compress the relatively large public key values to a short and fixed-length field. A message including a self-signature typically has the format $\{M.h(P_X)\}_{P_X^{-1}}.P_X$, i.e. consists of some payload M and $\Psi_X = h(P_X)$ jointly signed³ with the private key P_X^{-1} and concatenated with P_X . In this way, the receiver (which is the Location Server here) can use the public key provided to check the signature and see if the hash of the public key matches the corresponding value in the signed part. If so, he can be sure that the sender holds the private key related to both the public key and the pseudonym, without learning the identity of the sender. Of course, authentication can only be ensured in combination with other means. If self-signatures are used in isolation, any party can produce them and then mount, e.g., denial-of-service attacks.

Furthermore, this variant uses public key cryptography for securing the messages of the agreement phase and signing the location information message sent by the Location Server.

3.1 Protocol Specification

The overall protocol, in the usual Alice-Bob notation, reads as follows.

Agreement. $T \leftarrow \{LR\}_{K_T}.\{\{T.N_1\}_{K_T}.\Psi_{LR}\}_{K_{LR}^{-1}} - LR$
 $T \xrightarrow{\hspace{1.5cm}} \{\{N_1\}_{K_{LR}}.\Psi_T\}_{K_T^{-1}} \rightarrow LR$

This phase is secured using ordinary public-key encryption and signatures of T and LR , with the public keys K_T and K_{LR} as well as their private counterparts K_T^{-1} and K_{LR}^{-1} . In order to register with T and obtain T ’s pseudonym, LR sends to T its own identity LR , T ’s identity (as a redundancy that can be checked by T), a nonce N_1 (a “random” value used to ensure freshness for the authentication of T), as well as its own pseudonym $\Psi_{LR} = h(P_{LR})$. The name LR , used by T to identify LR , is encrypted with T ’s public key such that LS has no chance to learn the association of Ψ_{LR} with the real identity LR . Note that we cannot move the name LR inside the signed part, because T first needs to derive LR ’s public key from this name before being able to decrypt the signed part.

If T is willing to share its location (up to some granularity) with LR , it answers with the nonce just received and the pseudonym $\Psi_T = h(P_T)$. Note that the pseudonyms do not need to be encrypted, but just signed.

Policy. $T \xrightarrow{\hspace{1.5cm}} \{GR.\Psi_{LR}.\Psi_T\}_{P_T^{-1}}.P_T \longrightarrow LS$

T sends to LS its policy aka Privacy Rule, comprising the granularity GR and the pseudonyms of LR and T , in a self-signed way.

³ In practice, a message \mathcal{M} is signed with a key Y^{-1} by appending \mathcal{M} with the hash of \mathcal{M} decrypted with Y^{-1} , which we would denote by the term $\mathcal{M}.\{h(\mathcal{M})\}_{Y^{-1}}$, but for our purposes (assuming \mathcal{M} contains enough redundancy, and abstracting from other issues like computational efficiency) it suffices to use the simpler term $\{\mathcal{M}\}_{Y^{-1}}$ that avoids repeating \mathcal{M} .

Location. $T \longrightarrow \{TS.\{LI.\Psi_T\}_{K_{LS}}\}_{P_T^{-1}}.P_T \longrightarrow LS$

T informs LS about its current location, by sending, again in a self-signed way, a timestamp TS (which can be used to avoid replay attacks) as well as the Location Information LI (with maximal accuracy) and its pseudonym Ψ_T , both encrypted with the public key K_{LS} .

Information. $LS \longleftarrow \{\Psi_{LR}.\Psi_T.N_2\}_{P_{LR}^{-1}}.P_{LR} - LR$
 $LS \longrightarrow \{\{GR(LI)\}_{P_{LR}}.N_2\}_{K_{LS}^{-1}} \longrightarrow LR$

Finally, LR requests T 's current location by sending to LS , also in a self-signed way, its own pseudonym, T 's pseudonym, and a nonce N_2 . The nonce ensures freshness for the authentication of LS and identifies the answer expected from LS . Therefore LS does not need to echo Ψ_{LR} or Ψ_T .

LS matches the pseudonyms with its Privacy Rule table, which is used also for looking up the granularity GR specified by T . If found, LS replies with a message containing the location data and the nonce, where the location data is LI projected to the granularity GR and encrypted with P_{LR} . Since the encryption key P_{LR} and the nonce N_2 are not secret, such that anyone could construct such a message, LS has to sign the message with its private key K_{LS}^{-1} in order to authenticate itself to LR .

Our model does not support location updates by re-sending the Location message with new data. Therefore, replay protection for the authenticity of $GR(LI)$ is simple. If location updates are possible, one must do more to prevent replay attacks, namely use a timestamp. Since HPLSL does not support time, we include a pseudo-timestamp TS just as a reminder.

Furthermore, we do not explicitly model authorization (as opposed to authentication) of the Location Recipient in the Agreement sub-protocol, or authorization of the Target in the Policy sub-protocol. We simply take the worst-case assumption that they are authorized unconditionally.

3.2 Requirements Specification

The protocol has been designed to enjoy the following security properties, which (apart from the last two ones) are immediate formalizations of the requirements stated in RFC3693 [CMM⁺04].

- secrecy of LI and of the filtered version $GR(LI)$
- LR strongly authenticates LS on N_2
- LS weakly authenticates LR on P_{LR}
- LS weakly authenticates T on GR
- LR strongly authenticates T on $GR(LI)$
- LR strongly authenticates T on N_1
- T weakly authenticates LR on Ψ_{LR}

The phrase “ X weakly authenticates Y on Z ” means that X can be sure that its peer is indeed Y and the two parties use the same value Z . This the same as Lowe’s notion of *non-injective agreement* [Low97]. Strong authentication additionally entails replay protection, i.e. freshness of the agreement (or session)

between the two, and directly corresponds to Lowe's *injective agreement*. The term "*LR strongly authenticates T*" appears twice — the first instance referring to the authenticity of *GR(LI)* in the Information phase, the second one referring to the authentication of *T* in the Agreement phase.

The formalization of both the protocol and its security requirements in the specification language HLPSL can be found in the appendix as well as on-line at <http://www.avispa-project.org/library/self-signatures.html>.

3.3 Design Process and Analysis Results

In designing the protocol, we have been careful to fulfill the anonymity requirements, which unfortunately cannot be checked by the tools at hand. Moreover, we aimed at minimizing the number of message fields and the use of cryptographic operations, and we have tried to get as far as possible wrt. replay protection without adding extra message exchanges. For evaluating the large number of intermediate protocol versions wrt. structural correctness, confidentiality and authentication, the AVISPA Tool [AT-05] has proved very useful – we have been able to check these versions in an exploratory way, very quickly and easily.

For instance, in this way we have found that in the Location message, if we encrypt *LI* only (resulting in the message $\{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{P_T^{-1}.P_T}$), we get a man-in-the-middle attack against both authentication of *LR* and secrecy of *GR(LI)*, the trace⁴ of which is the following:

$$\begin{array}{l}
 i \leftarrow \{LR\}_{K_T}.\{\{T.N_1\}_{K_T}.\Psi_{LR}\}_{K_{LR}^{-1}} - LR \\
 T \leftarrow \{LR\}_{K_T}.\{\{T.N_1\}_{K_T}.\Psi_{LR}\}_{K_{LR}^{-1}} - i \\
 T \xrightarrow{\{\{N_1\}_{K_{LR}}.\Psi_T\}_{K_T^{-1}}} i \\
 T \xrightarrow{\{GR.\Psi_{LR}.\Psi_T\}_{P_T^{-1}.P_T}} i \\
 T - \{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{P_T^{-1}.P_T} \rightarrow i \\
 i(T) \xrightarrow{\{GR.h(P_j).h(P_i)\}_{P_i^{-1}.P_i}} LS \\
 i(T) - \{TS.\{LI\}_{K_{LS}}.h(P_i)\}_{P_i^{-1}.P_i} \rightarrow LS \\
 i(LR) \xrightarrow{\{h(P_j).h(P_i).N_2\}_{P_j^{-1}.P_j}} LS \\
 i(LR) \leftarrow \{\{GR(LI)\}_{P_j}.N_2\}_{K_{LS}^{-1}} \xrightarrow{\quad} LS
 \end{array}$$

The essence of this attack is that after the Target has sent its messages, the intruder can re-use the encrypted value $\{LI\}_{K_{LS}}$ intercepted from *T* and pose towards *LS* as both a Target and a Location Recipient, hijacking the original

⁴ In our setting, a *trace* is a kind of message sequence chart describing which messages are sent between which parties in which order. Due to the standard Dolev-Yao intruder model [DY83] which we employ, messages between honest parties are always sent via the intruder, denoted by *i*, who may decide to suppress, modify, or forward them to any party. Where the intruder poses as role *R*, we write *i(R)*. An *attack trace* is a trace leading to an *attack state*, i.e. a state of the parties involved (including the intruder) where one of the desired security properties is violated.

session. In this way, he can trick the Location Server to send to him, rather than to the legitimate Location Receiver, a copy of $GR(LI)$ encrypted with a public key P_j . Since P_j can be chosen freely by the intruder, he can decrypt the location data. To prevent this attack, we simply move Ψ_T inside the encryption, arriving at the message $\{TS.\{LI.\Psi_T\}_{K_{LS}}\}_{P_T^{-1}}.P_T$. Now any attempt to replay $\{LI.\Psi_T\}_{K_{LS}}$ in a different context does not fit because Ψ_T does not match with the hash of any public key for which the intruder has the corresponding private key needed for producing the self-signature for the Location message.

Actually, due to anonymity, the Location Server cannot authenticate the Target and the Location Receiver individually but only check the consistency of the self-signed messages received from the two. This also implies that the intruder can always pose as both these parties simultaneously, provoking the following degenerate exchange:

$$\begin{array}{l}
 i(T) \text{ ----- } \{GR.h(P_j).h(P_i)\}_{P_i^{-1}}.P_i \rightarrow LS \\
 i(T) \text{ ----- } \{TS.\{LI.h(P_i)\}_{K_{LS}}\}_{P_i^{-1}}.P_i \rightarrow LS \\
 i(LR) \text{ ----- } \{h(P_j).h(P_i).N_2\}_{P_j^{-1}}.P_j \rightarrow LS \\
 i(LR) \text{ <----- } \{\{GR(LI)\}_{P_j}.N_2\}_{K_{LS}^{-1}} \text{ ----- } LS
 \end{array}$$

Yet this is a useless attack (apart from wasting the Location Server's resources) because the intruder does not obtain anything interesting — he receives just the location data that he has provided himself. As soon as a faithful Location Recipient or Target is involved, correct authentication is guaranteed because the Location Server can check the self-signed messages of these two parties for consistency of the pseudonyms involved. This consistency, in turn, is guaranteed by the mutual authentication of the Location Recipient and the Target. Hence if one of these is authentic, then the other is, too.

4 Variant with Password and Certificate

Our second variant uses a pseudonym only for the Target and uses password for the Location Recipient. The Target authenticates itself to the Location Server not with a self-signature, but more stringently, with a certificate issued by a trusted third party. The Agreement phase is secured using a shared secret key (or any other form of a secure channel) rather than with public-key cryptography, such that both the Target and the Location Recipient depend on a public-key infrastructure (PKI) only for sending messages to the Location Server. In the Information phase, the Location Recipient provides a (secret) temporary key to be used for encrypting the location data. All this amounts to four major differences to our first variant.

4.1 Protocol Specification

The protocol is specified as follows.

Agreement. $T \xleftarrow{LR.\{T.N_1\}_{K_{TLR}}} LR \xrightarrow{\{PW_T.\Psi_T.N_1\}_{K_{TLR}}} LR$

This phase is secured with a secret key K_{TLR} shared between T and LR . In order to register with T and obtain the password, LR sends to T its own identity LR , T 's identity and a nonce N_1 . The name LR is sent in the clear in this case, because T needs a way to tell who it is talking to and select the right key K_{TLR} . This is fine as long as LS cannot intercept and link the first message with the anonymous request it receives in the Information phase. If T is willing to share its location with LR , it answers with the nonce just received, its pseudonym Ψ_T , and the password PW_T . Actually, signing the pseudonym would be sufficient, rather than encrypting it.

Policy. $T \xrightarrow{\{GR.\{PW_T\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}}.\{\Psi_T.K_T\}_{K_{CA}}} LS$

T sends to LS its Privacy Rule, comprising the granularity GR , the password (in encrypted form), and its pseudonym. All this is signed with T 's private key. The receiver LS does not know T 's identity, yet in order to check the signature, it is sufficient that T encloses a certificate — signed by a trusted third party called CA — stating the relation between the pseudonym and T 's public key. Of course, this means an extra overhead, which can be reduced in cases where LS is allowed to know the identity of the Target.

Location. $T \xrightarrow{\{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}}} LS$

T conveys to LS its current location, by sending, again signed with K_T^{-1} , its pseudonym Ψ_T and a timestamp TS along with the Location Information LI . Here only LI needs to be encrypted with the public key K_{LS} because LS can authenticate T independently of LR , using the certificate received in the Policy phase. Therefore the two attacks explained in section 3.3 are a priori not possible.

Information. $LS \xleftarrow{\{K_{LR}.PW_T.N_2\}_{K_{LS}}} LR \xrightarrow{\{GR(LI).N_2\}_{K_{LR}}} LR$

Finally, LR requests T 's current location by sending to LS a temporary key K_{LR} , the password related to T (which actually renders sending along T 's pseudonym unnecessary) and a nonce N_2 . All this is encrypted with LS 's public key.

LS matches the password (and T 's pseudonym, if sent nevertheless) with its Privacy Rule table. If found, LS replies with a message containing the location data and the nonce. Since the encryption key K_{LR} and the nonce N_2 have been kept secret, LR just needs to encrypt the location data and the nonce with K_{LR} in order to protect the location data and to authenticate itself to LR .

The two notes wrt. location updates and authorization that we have given for the other variant, apply also for this variant: location updates are not modeled, which simplifies replay protection, and concerning authorization we have taken the most pessimistic and simple assumption that any access is granted.

No other (spurious or actual) attacks on this protocol have been found.

The formalization of both the protocol and its security requirements in HLPSSL are available on-line at <http://www.avispa-project.org/library/password.html>.

5 Conclusion

We have proposed protocols that form the core of Geopriv services, meeting the challenge of anonymity despite authentication. This is the main novelty of our protocol design. We have made explicit the design considerations that we took, such that the protocols can easily be evaluated and re-used by others. We have emphasized that many variations of the protocols are possible and have exemplified two of them. Others would be interesting, too, for instance combining the strong certificate-based authentication of the Target given in our second variant with the self-signed pseudonym approach for the Location Recipient given in our first variant (which is more strict than the password-based approach). Implementors are free to choose among all those variations according to their preferences and side-conditions imposed by the application context.

During the protocol design, it proved very helpful to formalize the various versions of the protocol, as well as their intended properties, in a designated protocol specification language and to check with automatic tools whether the given requirements are fulfilled. Since this approach greatly reduces both time and effort, we believe that it should — and soon will — become the standard approach for crypto protocol design, be it academic or industrial.

Apart from the issues discussed, the model checkers do not find attacks. Since we have carefully reviewed our formalizations to validate that they faithfully describe the protocol design, and since the tools used are quite mature, we can be confident that in the proposed Geopriv core protocols there are no design flaws that can lead to attacks on confidentiality and authentication. Of course, vulnerabilities at the cryptographic or implementation level cannot be excluded with this approach, and the anonymity aspect has been checked only informally.

6 Outlook

As mentioned, for each of the four Geopriv sub-protocols, there are various orthogonal ways to implement them. Therefore, it would be nice to reflect the inherent modularity also during formal analysis, allowing to verify each of the variants of the different phases separately and then perform some compositional reasoning to arrive at the overall security properties, combining the common properties of the variants for each of the phases involved.

Moreover, there are generalizations where several Geopriv instances are combined on demand at runtime, to build a chain of Location Servers with the policies and the Location Information flowing along a chain of trust among them. For instance, a Local Location Server may immediately sense the location, pass it on via an intermediate Remote Location Server to a Home Location Server, which is the

only one the Target trusts. In this way, three Geopriv instances form a single higher-level instance. For analyzing such a scenario, a compositional reasoning technique is not only desirable, but actually indispensable because of combinatorial explosion.

Further complications can arise from a different sort of privacy: when policies depend on the Location Server and, for a given untrusted Location Server L , the part of their contents that is not relevant to L should not be visible for L .

Orthogonally, there is the issue of dynamic policies in the sense of Privacy Rules evolving over time, granting additional access and/or revoking access to location data of various targets to various receivers. This may involve additional pitfalls that would better be checked with the meticulousness of formal methods.

In a project related to AVISPA, we plan to do further research that will tackle all the issues mentioned above.

Acknowledgments. Initial versions of the protocol models described here were done by Lan Liu in her master's thesis [Liu05]. We thank Hariharan Rajasekaran and some anonymous referees for their comments on earlier versions of this paper.

References

- AH-03. The AVISPA project homepage. <http://www.avispa-project.org/>, 2003.
- AT-05. The AVISPA Tool. Available at <http://www.avispa-project.org/>, 2005.
- AVI05a. HLPSTL Tutorial: A Beginner's Guide to Modelling and Analysing Internet Security Protocols, 2005. Available at [AH-03].
- AVI05b. AVISPA User Manual, 2005. Available at [AH-03].
- CCC⁺04. Yannick Chevalier, Luca Compagna, Jorge Cuellar, Paul Hankes Drielsma, Jacopo Mantovani, Sebastian Mödersheim, and Laurent Vigneron. *A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols*, volume 180 of *Automated Software Engineering*, pages 193–205. Austrian Computer Society, Austria, September 2004.
- CMM⁺04. J. Cuellar, J. Morris, D. Mulligan, J. Peterson, and J. Polk. RFC 3693: geopriv requirements, 2004. <http://www.faqs.org/rfcs/rfc3693.html>.
- DMMP04. M. Danley, D. Mulligan, J. Morris, and J. Peterson. RFC 3694: Threat Analysis of the Geopriv Protocol, 2004. <http://www.faqs.org/rfcs/rfc3694.html>.
- DY83. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
- GWG06. IETF Working Group: Geographic location/privacy (geopriv), 2006. <http://www.ietf.org/html.charters/geopriv-charter.html>.
- Liu05. Lan Liu. Analyzing web service protocols with the AVISPA approach. M.Sc. thesis, Universität Karlsruhe and Siemens, 2005.
- Low97. Gavin Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 31–43. IEEE Computer Society Press, 1997.

A Formalization of the First Protocol Variant in HLPSL

```

role target(
  T, LS, LR : agent,
  K_T, K_LS, K_LR : public_key,
  H : hash_func,
  Snd_LR, Snd_LS, Rcv : channel(dy)) played_by T def=

local
  State : nat,
  N1 : text,
  P_T : public_key,
  Psi_LR : hash(public_key),
  LI, TS : text,
  GR : hash_func

init State := 1

transition

  1. State = 1 /\ Rcv({LR}_K_T.{T.N1'}_K_T.Psi_LR')_inv(K_LR)
  => State' := 3 /\ P_T' := new()
      /\ Snd_LR({N1'}_K_LR.H(P_T'))_inv(K_T)
      /\ witness(T, LR, lr_T_N1, N1')
      /\ wrequest(T, LR, t_LR_Psi_LR, Psi_LR')
% could be new transition here, but not done for efficiency
      /\ GR' := new() % chooses some granularity (accuracy)
      /\ Snd_LS({GR'.Psi_LR'.H(P_T')}_inv(P_T').P_T')
      /\ witness(T, LS, ls_T_GR, GR')
% could be new transition here, but not done for efficiency
      /\ LI' := new()
      /\ secret(LI', li, {T, LS, LR})
      /\ secret((GR'(LI')), filtered_LI, {T, LS, LR})
      /\ TS' := new()
      /\ Snd_LS({TS'}.{LI'.H(P_T')}_K_LS}_inv(P_T').P_T')

      /\ witness(T, LR, lr_T_filtered_LI, (GR'(LI')))
      /\ witness(LS, LR, ls_LR_P_LR, LS)

end role

```

```

role locationServer(
  T, LS, LR: agent, % but LS does not actually use identity of T and LR
  Psi_Table: (hash(public_key).hash(public_key).hash_func) set,
  K_LS : public_key,
  H : hash_func,
  Snd, Rcv : channel(dy)) played_by LS def=

local State : nat,
  P_T, P_LR : public_key,
  N2 : text,
  Psi_LR : hash(public_key),
  LI, TS : text,
  GR : hash_func

init State := 5

transition

  5. State = 5 /\ Rcv({GR'.Psi_LR'.H(P_T')}_inv(P_T').P_T')
  => State' := 7 /\ Psi_Table' := cons(Psi_LR'.H(P_T')).GR', Psi_Table)

  7. State = 7 /\ Rcv({TS'}.{LI'.H(P_T')}_K_LS}_inv(P_T).P_T)
  => State' := 9

```

```

9. State = 9 /\ Rcv({H(P_LR') . H(P_T). N2'}_inv(P_LR') . P_LR')
  /\ in(H(P_LR') . H(P_T). GR', Psi_Table)
  % uses Psi_LR and Psi_T to look up GR in the table
=> State' := 11 /\ Snd({(GR'(LI))}_P_LR'. N2'}_inv(K_LS))
  /\ wrequest(LS, T, ls_T_GR, GR') % delayed
  /\ wrequest(LS, LR, ls_LR_P_LR, P_LR')
  /\ witness (LS, LS, lr_LS_N2, N2') % to any LR!

end role

```

```

role locationRecipient(
  T, LS, LR      : agent,
  K_T, K_LS, K_LR : public_key,
  H              : hash_func,
  Snd, Rcv       : channel(dy) played_by LR def=

```

```

local
  State      : nat,
  N1, N2     : text,
  Psi_T      : hash(public_key),
  P_LR       : public_key,
  Filtered_LI : hash(text)

```

```

init State := 0

```

```

transition

```

```

0. State = 0 /\ Rcv(start)
=> State' := 2 /\ N1' := new()
  /\ P_LR' := new()
  /\ Snd({LR}_K_T. {{T.N1'}_K_T.H(P_LR')}_inv(K_LR))
  /\ witness(LR, T, t_LR_Psi_LR, H(P_LR'))

2. State = 2 /\ Rcv({{N1}_K_LR.Psi_T'}_inv(K_T))
=> State' := 8 /\ N2' := new()
  /\ Snd({H(P_LR).Psi_T'. N2'}_inv(P_LR) . P_LR)
  /\ witness(LR, LS, ls_LR_P_LR, P_LR)
  /\ request(LR, T, lr_T_N1, N1)
  /\ witness(LS, T, ls_T_GR, LS)

8. State = 8 /\ Rcv({{Filtered_LI'}_P_LR.N2}_inv(K_LS))
=> State' := 10 /\ request(LR, T, lr_T_filtered_LI, Filtered_LI')
  /\ request(LS, LS, lr_LS_N2, N2)

```

```

end role

```

```

role session(T, LS, LR      : agent,
  K_T, K_LS, K_LR      : public_key,
  H                    : hash_func,
  Psi_Table            : (hash(public_key).hash(public_key).hash_func) set) def=

```

```

local STLR, STLS, RT, SLR, RLR, SLS, RLS: channel(dy)

```

```

composition

```

```

  target      (T, LS, LR,      K_T, K_LS, K_LR, H, STLR, STLS, RT)
  /\ locationServer (T, LS, LR, Psi_Table, K_LS,      H, SLS, RLS)
  /\ locationRecipient(T, LS, LR,      K_T, K_LS, K_LR, H, SLR, RLR)

```

```

end role

```

```

role environment() def=

local Psi_Table: (hash(public_key).hash(public_key).hash_func) set
  % shared between all instances of LS

const li, filtered_LI,
  ls_T_GR,
  lr_T_N1,
  t_LR_Psi_LR,
  ls_LR_P_LR,
  lr_LS_N2,
  lr_T_filtered_LI      : protocol_id,
  t, ls, lr             : agent,
  k_T, k_LS, k_LR, k_i  : public_key,
  h                     : hash_func

init Psi_Table := {}

intruder_knowledge = {t, ls, lr, k_T, k_LS, k_LR, k_i, inv(k_i), h}

composition

  session(t, ls, lr, k_T, k_LS, k_LR, h, Psi_Table)
%  /\ session(t, ls, lr, k_T, k_LS, k_LR, h, Psi_Table)
%  % repeat session to check for replay attacks

  /\ session(i, ls, lr, k_i, k_LS, k_LR, h, Psi_Table)
%  % It does not make much sense to let the intruder play the role of T
%  % since then the intruder knows its location information anyway.

  /\ session(t, ls, i, k_T, k_LS, k_i, h, Psi_Table)
%  % It does not make much sense to let the intruder play the role of LR
%  % since then the intruder is allowed to know the (secret) location of T.

end role

-----

goal

  secrecy_of li, filtered_LI

% strong authentication and integrity of the Location Information,
% (including replay protection):
  authentication_on lr_T_filtered_LI

% the Location Recipient Location authenticates the Location Server:
  authentication_on lr_LS_N2

% the Location Server (weakly) authenticates the Location Recipient:
  weak_authentication_on ls_LR_P_LR

% weak authentication and integrity of Granularity
  weak_authentication_on ls_T_GR

% additional authentication goals, not in RFC3693:
  authentication_on lr_T_N1
  weak_authentication_on t_LR_Psi_LR

end goal

-----

environment()

```


Delegation in a Distributed Healthcare Context: A Survey of Current Approaches

Mila Katzarova and Andrew Simpson

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD
United Kingdom

Abstract. The development of infrastructures to facilitate the sharing of data for healthcare delivery and research purposes is becoming increasingly widespread. In addition to the technical requirements pertaining to efficient and transparent sharing of data across organisational boundaries, there are requirements pertaining to ethical and legal issues. Functional and non-functional concerns need to be balanced: for resource sharing to be as transparent as possible, an entity should be allowed to delegate a subset of its rights to another so that the latter can perform actions on the former's behalf, yet such delegation needs to be performed in a fashion that complies with relevant legal and ethical restrictions. The contribution of this paper is twofold: to characterise the requirements for secure and flexible delegation within the emerging distributed healthcare context; and to evaluate existing approaches with respect to these requirements. We also suggest how some of these limitations might be overcome.

1 Introduction

The linking and aggregation of data to facilitate distributed healthcare delivery and research is becoming increasingly widespread. For example, the UK government has invested in a National Programme for Information Technology (NPfIT) in the National Health Service, which promises to deliver (amongst other things) electronic records, electronic prescription of drugs and electronic booking of appointments, all of which will be underpinned by an IT infrastructure [13]. Research has been characterised as a 'secondary use'.

Within Europe, the evaluation of *grid computing* [7] technologies to support healthcare research and delivery has been undertaken within a number of projects. Potential benefits of this emerging technology include large-scale sharing of information between institutions to allow distributed data analysis. In this respect, the EU HealthGrid initiative [11] is attempting to ensure that technological advances developed by the grid computing community benefit healthcare research and delivery, and—more importantly—that the requirements associated with applications in these areas influence the direction of these advances.

The context of the work described in this paper is being undertaken within the context of the NeuroGrid project [10], which is developing an infrastructure

to support collaboration between clinicians and researchers. It is intended that NeuroGrid will allow algorithms and data management procedures to be made more accessible and interoperable: current neuro-imaging research is typically characterised by small studies carried out in single centres, which leads to limited possibilities for data and algorithm sharing. To this end, the NeuroGrid project is seeking to provide a common grid-based infrastructure for data and algorithm sharing for three clinical exemplars. Although each exemplar aims to solve problems unique to its branch of neuro-science, all of the exemplars have use cases and requirements in common: with the need for each site being capable of temporarily delegating access to its data to users at other sites being one.

The development of an appropriate delegation mechanism requires the striking of a balance between functionality—in maximising the use of such data to support research that may lead to improved healthcare—and ethical and legal concerns—ensuring that the rights of the participants are respected. In [15], a vision for secure Grid-enabled healthcare within the United Kingdom was outlined. (We shall base the remainder of our discourse upon the situation in the United Kingdom as it is within this context that the work described is being undertaken.) Within the United Kingdom, health grid research projects are obliged to adhere to the principles of the Data Protection Act of 1998 [6], which include “personal data processed for any purpose or purposes shall not be kept for longer than is necessary for that purpose or those purposes”—which is, of course, particularly pertinent with respect to delegation. Of course, if any health grid were to be successfully deployed within a healthcare environment in the UK, it would be obliged to take into consideration requirements pertaining to patient data. The National Health Service comprises a number of *hospital trusts*, each of which is an independent legal entity. Each hospital trust is legally responsible for the data held at its sites: this data is released only with respect to the principles of the Caldicott Guardian, which include [4] “access to patient-identifiable information should be on a strict need to know basis: only those individuals who need access to patient-identifiable information should have access to it, and they should only have access to the information items that they need to see.”

The structure of the remainder of this paper is as follows. In Section 2 we identify the requirements for a delegation mechanism within a decentralised healthcare context. In Sections 3–6 we provide a survey of current delegation mechanisms and evaluate them with respect to our requirements. Although various methods have been proposed, as we shall see none of them are guaranteed to provide a solution that will satisfy our requirements. In Section 7 we outline areas of future work with a view to satisfying these requirements before, in Section 8, summarising the contribution of this paper.

2 Requirements

To be able to evaluate possible mechanisms for achieving secure delegation within a distributed healthcare context, we first need to identify what security requirements such a mechanism should satisfy. Some of these requirements are derived

by the need for compliance with the Data Protection Act, while others are driven by the fact that each site (for our purposes, a site is a hospital or research institute) is legally responsible for its data and may impose unique constraints upon access to it. Further, the transfer of such data (within the healthcare delivery context, if not the research context) can only happen in limited circumstances as characterised by the principles of the Caldicott Guardian.

We do not restrict ourselves to those requirements that pertain only to research—we consider that a healthgrid should be developed with both healthcare delivery and research in mind. It should also be noted that we do not claim that these requirements are exhaustive: we restrict ourselves to functional requirements. Non-functional requirements such as performance, scalability and interoperability—although vital—are outside the scope of this paper.

2.1 Least-Privilege Delegation

Let us consider an example in which Doctor X would like to delegate permissions associated with access to MRI data to Doctor Y from a different hospital, with the intention that Doctor Y can review the scan and provide an opinion on it.

In this scenario, we call Dr X the *delegater* and Dr Y the *delegatee*. When delegating, the delegater would like to delegate only those rights which are absolutely necessary to complete a certain task. *Overdelegating* is not only undesirable, but it can also be quite dangerous. For example, consider a situation in which the delegater does not have the possibility to delegate read permissions to only one particular dataset from a pool of patient records. In this case, the delegater will have to delegate the entire set of his rights, i.e., permissions embracing the whole pool of patient records. This will allow the delegatee to access not only the desired patient information, but also records pertaining to other patients. This violates both the Data Protection Act and the principles of the Caldicott Guardian. Least-privilege delegation is, therefore, a key requirement for us.

2.2 Revocation

Another important requirement to impose on a potential delegation mechanism is that it should provide the ability for the delegater to revoke credentials whenever he or she thinks this is appropriate. Such revocation may be driven by the fact that credentials have been compromised or due to fact that the task has been completed. In the healthcare context, this requirement is driven by the following principle of the Data Protection Act: “personal data processed for any purpose or purposes shall not be kept for longer than is necessary for that purpose or those purposes.”

A simple revocation solution can be adopted in applications where security considerations associated are not so critical, i.e., the confidentiality of the data is not of a high priority and there are no ethical requirements (with the field of High Energy Physics being an appropriate example). In such situations, this

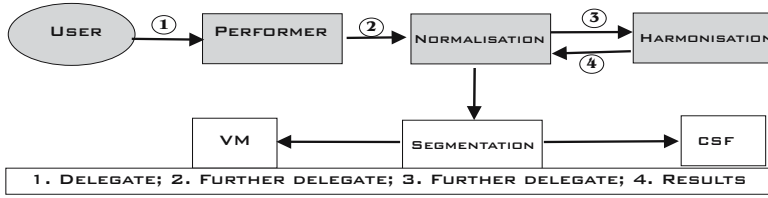


Fig. 1. Structural MRI analysis data flow

may be achieved by limiting the lifetime of the credentials and causing an automatic revocation after this period. Unfortunately, this cannot be applied in the healthcare domain as basic ethical constraints will be violated.

2.3 Onward Delegation

In many cases, multiple services may need to be iteratively invoked in order for a job to be completed. As an example, we may consider the following processing data flow performed within NeuroGrid.

Figure 1 provides an abstract view of the workflow for MRI structural dataset processing. *Performer* is a process which has been delegated the necessary rights to complete the task. The first step of the process is the *normalisation* (the process of unifying images taken from different types of scanners) of data. Part of *normalisation* in our example is *harmonisation*, which has to be invoked by the normalisation service. Consequently, necessary rights need to be transferred to the harmonisation service by the normalisation service. This requires delegation to be applied multiple times and gives rise to a number of crucial security questions.

Of course, it may be the case that the harmonisation service is not in the same domain as the normalisation service. For example, this may be a service provided by a different research institute and is covered by a completely different Certification Authority. The normalisation service, on the other hand, should be able to further delegate rights only in the case where a user whose rights are being used trusts the corresponding normalisation service. Ideally, information about the trusted entities should be included somehow and passed forward throughout the lifetime of delegation credentials.

2.4 Dynamically Changing Credentials

It must be possible to allow users to constrain the permissions they delegate based on some dynamic considerations. A typical use case is the scenario where a doctor delegates permissions to sensitive and non-sensitive data to another doctor at the same time. The non-sensitive data does not contain any patient-related information and the delegatee can perform multiple access for a long period of time. However, the delegator would also like to allow access to sensitive data but under certain conditions. These may include constraints which limit the access to the data to only once or for a limited period of time, or may oblige

the delegatee to erase any copies of the data created locally as a consequence of access to it.

Furthermore, as a consequence of the potentially dynamic nature of a grid, credentials often become obsolete. For example, users move and remove files, services stop working and hardware breaks down. In general, there is no insurance that resources which have recently existed are still available or that services previously running are still accessible. This problem arises from the fact that each organization is in possession of its own resources and further down the hierarchy, each user within the organization is in charge of his own resources. This requires immediate measures, such as, for example, a URL update of the file which has been replaced.

The above examples serve to illustrate the clear need for introducing dynamism of the credentials we are using to achieve delegation. *Hard-coded* credentials would not be convenient in this case.

Having defined a number of security requirements that a delegation mechanism should satisfy, we now survey some of the most popular current delegation technologies and consider the extent to which each complies with our requirements.

3 Proxy Certificates

X.509 proxy certificates [19] are a standard way of providing restricted proxying and delegation within a PKI-based authentication system and, currently, this approach is the most popular and most widely deployed approach to delegation within grid-based systems. Proxy certificates allow an entity holding a standard X.509 public key certificate to delegate some or all of its privileges to another entity, which may or may not hold X.509 credentials at the time of delegation. Once acquired, a proxy certificate is used by its bearer to authenticate and establish secured connections with other parties in the same manner as a normal X.509 end-entity certificate.

Proxy certificates use the format prescribed for X.509 public key certificates [12] and serve to bind a unique public key to a subject name, as a public key certificate does. However, unlike a public key certificate, the issuer (and signer) of a proxy certificate is identified by a public key certificate or another proxy certificate rather than a certification authority (CA) certificate. This approach allows proxy certificates to be created dynamically without requiring the normally heavyweight vetting process associated with obtaining public key certificates from a CA.

Onward delegation could be achieved via a recursive chain of proxy certificates. The first one is signed with the private key corresponding to the public key certificate of the issuer and each other down the chain is signed with a freshly generated by the next delegater (which appears as a delegatee in the previous step of the delegation chain) private key. The delegated rights are encoded in the Proxy Certificate Information extension (PCI) which also contains the allowed depth of a delegation chain. Proxy certificates, based on a mechanism described

in [9], have been deployed in a number of grid-based projects, with the EU Data-Grid [5] and Enabling Grid for E-sciencE (EGEE) [1] projects being pertinent examples. However, this scheme is inherently weak in terms of security.

First, if restriction is required then the delegation policy has to be specified in the proxy certificate. This solution is not a panacea as it raises tricky questions about how to specify the delegation policy, particularly if there are multiple resources involved. Note also that the delegation policy would have to specify whether onwards delegation was allowed, which rights could be delegated onwards, and to whom. This introduces unnecessarily complicated proxy credentials, in many cases too large as well, which have to be carried throughout the entire delegation chain. The more precise we would like to be in specifying the minimal rights the more complicated and large this credentials become.

Furthermore, revocation is not possible in this approach. All delegates are expected to destroy the secret component of their delegation key pair, to protect against subsequent compromise of any of the agents in the chain. However, this mechanism provides no protection against compromise of an active delegatee and it requires all delegates to be trusted to actually destroy their secret keys when finished. By limiting the lifetime of the proxy certificate a certain degree of security could be provided, but this does not provide the delegator with insurance that their rights will not be violated and also requires user intervention in the case of long lifetime jobs where the proxy certificate will have to be renewed.

Although onward delegation can be achieved in a relatively straightforward fashion using proxy certificates by simply creating a new proxy certificate for each stage of the chain, a problem may arise in the trust relationship between the participants in the chain. It is desirable that trust information is passed to all delegators in a delegation chain; although, proxy certificates do offer such a possibility, they do not provide a flexible way for update of this information. Considering the dynamic nature of grid-based environments, there is a potential risk of resources being replaced/removed and, consequently, the related credentials becoming obsolete. In such a situation, the authorization information in the proxy certificate has to be updated. The only possible of achieving this is via a recreation of the certificate. This, of course, will require user intervention and fails to comply with the single sign-on requirement.

Although proxy certificates seem to be a compact and deployable solution, their security (lack of revocation) and scalability (static authorization policy encoded in the certificate) drawbacks make them a potential source of security weaknesses in the widely-deployed Public Key Infrastructure.

4 Call-Backs

In [8] a set of protocols that are intended to solve delegation-related problems in computational grids is proposed, with the architecture being based on the notion of user and resource proxies. A user proxy (UP) is a session manager process given permission to act on behalf of a user for a limited period of time. The user

proxy acts as a representative of the user with its own short term credentials. A resource proxy (RP) is an agent used to translate between inter-domain security operations and local intra-domain mechanisms, and serves as the interface between the grid security architecture and the local security architecture.

The architecture aims to specify protocols for implementing the following typical scenario [8]: “A user (U) would like to run a job and delegates his rights to a process, called User Proxy (UP) to execute the job on his behalf. When UP requires access to a resource, it will first determine the identity of the Resource Proxy (RP) for that resource. It then issues a request to the corresponding RP and if the request is successful, the resource is allocated and a process is created on this resource.”

This is achieved by specifying a set of protocols defining actions between users (U), user proxies (UP) and resource proxies (RP). The first protocol is associated with the creation of a user proxy. When created, the UP is supplied with the necessary credentials to execute the job. The second protocol allows a user proxy to obtain resources so as to run processes. The user proxy communicates with a resource proxy, which allocates resources from some pool. The third protocol enables a process P, acting on behalf of a user U with user proxy UP, to obtain some other resource; thus the user proxy delegates rights to the process.

All resource allocation goes through the UP, so it is straightforward to ensure that resources are allocated in a manner that is consistent with the wishes of the user. The UP can be supplied with the necessary authorization information and allow access in accordance with it. The third protocol enables a process P, acting on behalf of a user U, with user proxy UP, to obtain some other resource; thus the user proxy delegates rights to the process.

Unfortunately this mechanism lacks scalability, primarily due to the fact that all resource allocations must go through the user proxy. This could potentially cause the user proxy to become a bottleneck in the system, particularly on large jobs, where many processes are to be acquired.

Another potential drawback of this approach is the problem of determining the minimal set of rights. Take, for example, a solution in which a user's job needs to be able to authenticate on behalf of the user to facilitate access to a mass storage system to store the results of a long computation. Even though the user may have access to the entire storage facility, this particular job requires only a limited amount of disk space to collect the results. In other words, there is no need to delegate the complete set of rights the user possesses. In callback delegation, no mechanism for precisely defining only the necessary rights has been suggested and this again implies delegation of rights to a much bigger set of resources than is usually required. In addition to this, no dynamic rights changes have been held in this regard either. An unexpected failure of the storage, rendering it temporarily unavailable, will prevent the job from storing the results. Consequently, the execution of the job will be resumed only after the storage appears available again or until permissions for a different resource have been granted, i.e., the delegator has to repeat the transfer of rights.

Even though one of our requirements (revocation) is satisfied, the other three (minimal set of rights, onward delegation and dynamic credentials) are not. This makes the call-back mechanism suitable for smaller, centralized distributed systems, where there is a local control over the resources. Unfortunately, it will certainly fail to be secure in a larger trans-organisational network where there is not quality insurance and trust relationships between entities can be rather unstable.

5 XML-Based Approaches

5.1 SAML

Security Assertion Markup Language (SAML) [3] is an OASIS standard for a framework for exchanging authentication, subject attribute and authorization information that is based on XML. While the current SAML standard lacks delegation capabilities, an extension that supports restricted onwards delegation has been proposed [18]. It is this extension that we consider here.

The proposed framework, which contains three XML-based components—delegation assertions, protocol requests and protocol responses—is based on *Attribute* statements and utilizes the SAML binding specification which defines protocol bindings for the use of request-response messages; this particular implementation uses the SOAP binding. Usually, a user delegates his rights to a web portal, S_1 , with S_1 then delegating the user's rights to another web service, S_2 , and so on (ending with S_{n-1} delegating to S_n). The user can then submit a job to S_1 and go offline. After that, any entity S_i ($1 \leq i \leq n$) can access the web service W and act on the user's behalf by presenting the delegated SAML assertion chain. The delegation from the user to S_1 constitutes direct delegation while any delegation from S_i to S_{i+1} ($1 \leq i \leq n$) is considered indirect.

The delegated rights (which do not describe which resources the delegatee is allowed to access, but rather are concerned only with the issue of whether the delegatee can further delegate the full set of delegator's rights to another entity) are described using special delegation SAML assertions. The delegation request and response messages conform to the SAML request and response protocol. The delegatee signs a delegation request to the delegator. The delegator uses an included *signature* element to authenticate the request.

This solution allows an entity to grant permissions to another in order for the latter to act on the former's behalf. The provided mechanisms for SAML request and response exchange and assertion validation ensure the successful and secure transfer of the delegated credentials. In addition to this, its applicability to web services-based technologies can be considered an advantage of this solution as it adds to its deployability. However, this approach fails to provide the issuer of the delegated assertions with the ability to restrict access to resources which are not associated with the action under consideration.

Recall our example where Dr X wishes to delegate permissions to Dr Y. As already stated, a critical requirement when delegating is the ability to provide

the issuer (Doctor X) with the means of constraining the delegated rights as much as possible. Moreover, in many cases, the delegator is obliged to restrict the access to certain resources (in this case, patient data). As we saw in the technical description of this approach, a SAML assertion contains only information concerning whether a delegation chain could be allowed or not (under the *RIGHT* field). In other words, Dr X cannot precisely express his wish to allow access only to the relevant file. Instead, Dr X must delegate the entire set of his permissions to Doctor Y.

Moreover, although it is possible to define whether the delegation can be further expanded to a chain, again, there is no information about who can participate in this chain. In the case when Dr X is willing to allow Dr Y to grant access to other colleague, the approach does not offer a possibility to define to who else the permissions could be transferred. If the *RIGHT* field is set to *Full*, then Dr Y is permitted to further delegate to any other subject. This may result in delegating access rights to a doctor who is located in a different country, potentially violating one of the principles of the Data Protection Act.

Finally, a secure revocation mechanism is not offered in the architecture. Instead, it suggests an approach similar to that adopted by proxy certificates, i.e., once created, the SAML assertions cannot be revoked before their expire date. We have already discussed the numerous disadvantages and hidden security threats when allowing a delegation method that does not support revocation *on-demand*, all of which are relevant here.

5.2 XACML

eXtensible Access Control Markup Language (XACML) [2] defines a core schema and corresponding namespace for the expression of authorization policies in XML against objects that are also identified in XML. XACML is emerging as a *de facto* standard due to its flexibility, extensibility and expressiveness.

Attempts at extending XACML to deal with delegation have been limited to achieving administrative delegation (see, for example, [17] and [14]). The commonly accepted approach consists of simply adding the necessary rules, i.e., changing the access policy, which will allow the delegatee to gain access to desired resources. Consequently, changing the policy back to its initial state has the effect of revoking the delegated rights.

Only the necessary permissions can be added to the access policy in an XACML policy document. As is the case for other approaches, this introduces concerns about the way of ensuring that only the necessary rights have been delegated. Revocation can be performed by changing the policy document back to what it was before the delegation occurred. A delegation chain will be impossible to be achieved here. If permissions concerning a particular participant has been revoked, then there is no way of revoking rights delegated to other entities down the delegation chain. Again, changing privileges could be performed by changing the policy document.

Very often, a doctor responsible for that particular patient would like to delegate temporary rights to an administrative assistant to perform some necessary

updates on the the record. In order for this to be achieved, the hospital policy has to be changed, i.e. the administrative assistant has to be allowed access to the record. In other words, the whole policy document has to be rearranged in order for such a simple delegation operation to be performed. In reality, access policies tend to be very complicated, and appropriate tool support to aid readability will be necessary in order for the system administrators to be able to adjust the policy to the delegation needs. However, even providing such tools would not be of a great help as in many cases it will be impossible for a human being to estimate, by just reading through the policy, how any changes will affect the access control system in overall. Certainly, some automatic correctness checks need to be applied, in order to ensure a particular delegation effect without creating side effects.

The second problem is scalability. If delegation is not the primary action performed within a certain access system, and consequently it does not appear to happen too often, than the policy and policy sets are not going to be changed as often. Consider, though, a data store which is used in collaboration between many research groups and upon which researchers are frequently running computational jobs utilizing datasets. These jobs are performed by special services which conduct the execution on behalf of users, and are required to be delegated with the necessary permissions to do so. The number of times the access policy of the storage will have to be changed will be significant. In addition, as discussed in the previous paragraph, all changes need to be verified and checks proving the absence of side effects will have to be performed upon each change. Clearly, such an approach would not scale.

6 Role-Based Delegation

In [20] a role-based delegation framework is presented, which includes: a delegation model that adapts the role-based delegation model RDM200 [21] and a rule-based language for specifying and enforcing delegation and revocation policies. The model is an extension of RBAC96 [16], and addresses the user-to-user delegation supporting role hierarchies and multi-step delegation in role-based systems. The paper defines a *delegation relation*, which attempts to address the relationship among deferent components involved in a delegation.

A declarative rule-based language is adopted to specify and enforce policies that allow users to delegate their roles, and consequently, revoke delegated roles. The system architecture is based on a delegation agent, which authorizes delegation and revocation requests from users by applying derivation authorization rules and processes delegation and revocation transactions on behalf of users. The result of an authorized delegation or revocation is sent and saved to a role-based access control database.

This approach provides a simple and intuitive way of achieving delegation. Its use of the role-based approach for access control minimizes the need for complicated access control policies to be composed by the user. Furthermore, an appropriate mechanism for revocation which corresponds to the chosen delegation approach has

been employed and secure onward delegation can be easily achieved. The method, however, fails to meet the least-privilege requirement, and is arguably difficult to apply within a decentralized system. Furthermore, a dynamic environment with frequent policy changes would have the potential to result in inefficiency.

The actual transfer of rights here is achieved by assigning roles to the delegates, which may give the delegatee a wider range of access control rights than absolutely necessary. In many cases, the delegatee needs to access a single file and for a very short period of time; as such, the role-based approach is inappropriate.

The mechanism is presented in the context of a single institution, with future consideration of an extension to support delegation across institutions suggested. As the authors suggest, a sensible mapping between roles across institutions would be necessary; this may not always be possible for virtual (and potentially short-lived) collaborations.

There is also the potential for performance drawbacks. The authors consider the case where the delegator knows which rights he or she would have to delegate in order for a task to be completed. This, however, does not cover the case where the delegator would like to perform a more complicated task and would like to delegate the execution of this task to a service. The service may have to delegate and revoke multiple times rights to another (or several) services in order to complete the job. This will require an update on the local policy of each participating institution, leading to afore-mentioned performance problems and side-effects.

7 Towards Secure Delegation in a Distributed Context

In the previous four sections we surveyed potential mechanisms for achieving delegation in a distributed environment, with none of the mechanisms satisfying our requirements of Section 2. In this section we provide a (necessarily) brief overview of initial steps towards the development a delegation architecture that satisfies our requirements.

The least-privilege delegation requirement can be considered as two separate aspects: determining the minimal set of rights to be delegated and representing this minimal set of rights. The problem of determining the minimal set of rights in a healthcare context will be task-dependent. In the simple case where the resources to which access is required are known to the delegator in advance, this requirement will pertain to providing the necessary tooling which will allow the user to simply describe the delegated rights; however, in the case where a user submits a job, determining the minimal set of rights the job will require throughout its lifetime is a significantly more complicated problem. Very frequently the user is not, and should not, be aware of the intermediate results of the job. Hence, he or she will not be able to precisely define the minimal set of rights and the delegation architecture should provide the automated means to do so without burdening the user with unnecessary information and actions. The

approach we have undertaken in this direction attempts to determine the minimal set of rights via examining the workflow description of the job in question.

After we have identified the minimal set of rights, we need to be able to present them in a standard way. For this purpose, we use XACML, which is emerging as the *de facto* standard for representing access control policies. Furthermore, by utilising this specification, we not only make use of the expressiveness it provides, but also of the standardisation of how the evaluation of access control decisions is to be performed. This helps us to move away from our delegation architecture the actual evaluation of the access control requests.

Once we have determined the minimal set of rights to be delegated, and have the means to describe them in a standard and efficient way, we need a mechanism via which it is possible to transform these rights to the delegatee(s). Furthermore, the mechanism should provide to appropriate means for further delegating these rights and revoking them at any point. We can consider two classes of mechanism: those which require local policy changes to achieve the desired delegation effect; and those which rely on external policy holders of the delegated rights (a la proxy certificates). The latter approach results in a more flexible architecture and does not introduce the potential for performance overheads. The development of a prototype—based on the concept of a trusted entity which will be supplied with delegation policy and will serve requests from resource accompanied by a delegation ticket—is ongoing.

Finally, the dynamic nature of the environment implies the need for a flexible means for updating credentials. It is hoped that later versions will allow such changes to be performed in a relatively seamless fashion.

The above four strands of activity represent a summary roadmap for the delivery of an infrastructure that satisfies our four requirements.

8 Conclusions

The development of infrastructures to facilitate the sharing of data for healthcare delivery and research purposes is becoming increasingly widespread. Delegation of permissions in such systems is vital with respect to maximising functionality, and is important that such functionality is delivered in a secure fashion and is consistent with the necessary legal and ethical frameworks.

The contribution of this paper is threefold. First, we have described the requirements for a delegation mechanism within a distributed healthcare research and delivery context, with these requirements being derived from our experiences within the NeuroGrid project (as well as other projects) and the relevant ethical and legal frameworks. Second, we have evaluated current delegation with respect to these requirements. Third, we have indicated initial steps towards the delivery of a delegation framework that satisfies our requirements.

Acknowledgments. The authors wish to thank the anonymous reviewers for their helpful comments. Mila Katzarova acknowledges the support of the NeuroGrid project.

References

1. EGEE project. <http://egee-intranet.web.cern.ch/>.
2. Oasis eXstensible Access Control Markup Language Committee, XACML V2.0. www.oasis-open.org/committees/.
3. Oasis Security Services Technical Committee, SAML V2.0. www.oasis-open.org/committees/.
4. The Caldicott Report. www.hpa.org.uk/confidentiality/caldicott.htm, Dec 1997.
5. Datagrid project. <http://eu-datagrid.web.cern.ch/eu-datagrid/>.
6. Data protection act 1998. The Stationery Office Limited, London, 1998.
7. I. Foster and C. Kesselman, editors. *The Grid: Blueprint For A New Computing Infrastructure*. Morgan Kaufmann, 1999.
8. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *5th ACM Conference on Computer and Communications Security*, pages 83–92, 1998.
9. M. Gasser and E. McDermott. An architecture for practical delegation in a distributed system. In *IEEE Symposium on Research in Security and Privacy*, pages 20–30, May 1990.
10. J. Geddes, S. Lloyd, A. C. Simpson, M. Rossor, N. Fox, D. Hill, J. Hajnal, S. Lawrie, A. McIntosh, E. Johnstone, J. Wardlaw, D. Perry, R. Procter, P. Bath, and E. Bullmore. Neurogrid: Using grid technology to advance neuroscience. In *18th IEEE Symposium on Computer-Based Medical Systems*, pages 570–572, 2005.
11. Healthgrid white paper. <http://whitepaper.healthgrid.org>.
12. R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, Apr 2002.
13. M. Humber. National Programme for Information Technology. *British Medical Journal*, 328(7449):1145–1146, 2004.
14. G. Navarro, B. Sadhigi-Firozabadi, E. Rissanen, and J. Borrell. Constrained delegation in XML-based access control and digital rights management standards. In *Proceedings of Communication, Network and Information Security*, Dec 2003.
15. D. J. Power, E. A. Politou, M. A. Slaymaker, and A. C. Simpson. Towards secure grid-enabled healthcare. *Software: Practice and Experience*, 35(9):857–871, 2005.
16. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.
17. L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozibadi, and O. Mulmo. Policy administration control and delegation using XACML and delegent. In *6th IEEE/ACM International Workshop on Grid Computing*, Nov 2005.
18. J. Wang, D. D. Vecchio, and M. Humphrey. Extending the security assertion markup language to support delegation for web services and grid services. In *IEEE International Conference on Web Services (ICWS 2005)*, Jul 2005.
19. V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *3rd Annual PKI R&D Workshop*, 2004.
20. L. Zhang, G. J. Ahn, and B. T. Chu. A role-based delegation framework for healthcare information systems. In *SACMAT*, pages 125–134, 2002.
21. L. Zhang, G. J. Ahn, and B. T. Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.

Managing Information Systems Security: Critical Success Factors and Indicators to Measure Effectiveness

Jose M Torres, Jose M Sarriegi, Javier Santos, and Nicolás Serrano

Department of Industrial Management Engineering, TECNUN, University of Navarra, Paseo
Manuel Lardizabal 13, 20018 San Sebastian, Spain
{jmtorres, jmsarriegi, jsantos, nserrano}@tecnun.es

Abstract. For how long can a business remain without its information systems? Current business goals and objectives highly depend on their availability. This highly dynamic and complex system must be properly secured and managed in order to ensure business survivability. However, the lack of a universally accepted information security critical factors' taxonomy and indicators make security management of information systems (SMIS) a tough challenge. Effective information security management requires special focus on identifying the critical success factors (CSFs) when implementing and ensuring SMIS. The purpose of this paper is to share a group of 12 CSFs identified in the current information security literature as well as a set of 76 indicators which are easy to calculate and attempt to provide valuable information to organizations seeking information security level measurements.

Keywords: Information systems, security management, critical success factors and indicators.

1 Introduction

Organizations, regardless of their size, are adopting Information Systems (IS) at a fast tempo in order to be more competitive. They have realized all the advantages that IS interconnections bring to organizations. This new way of communicating and doing business has placed information as one of the most critical assets for the majority of today's organizations.

The fast technology acquisition and IS "openness" increases systems' complexity and dependency. These two factors plus the ever increasing interruptions of critical business systems, uninvestigated security incidents, gaps in user awareness and the sophistication of threats make current business reactive security strategies highly risky and irresponsible approaches [1].

Security management of information systems (SMIS) challenges cannot be addressed in isolation. However, the lack of a universally accepted information security (infosec) framework, theories or tendencies and the absence of ways to measure the effectiveness of implemented infosec controls restrain organizations from identifying the real mechanisms that control information security behaviors [2]. As a result, SMIS has been understood so far as the set of point solutions (patching and

fixing breaches) instead of adopting more preventive and dynamic information security strategies.

Effective information security requires special focus on identifying the critical success factors (CSFs) for SMIS implementation and maintainability¹. In addition, information security indicators² should be implemented and analyzed in order to measure SMIS' effectiveness and better allocate security resources. This way, organizations could align information security with business goals and improve future security strategies, investments, and policy's enforcement.

Throughout this paper, the endlessly discussed but still vague information security definition is presented. However, the real purpose of this research is to point out on the one hand, which CSFs should be looked at in order to achieve robust and optimal SMIS's design, implementation and maintainability tasks. On the other hand, share a set of 76 indicators, identified in the current infosec literature, which could provide relevant information for organizations seeking ways to measure SMIS effectiveness. Certainly, there could be more complex and accurate ways to measure information security. However, practitioners' demand indicates the current need to find simple and fast ways to calculate organizational information security levels.

2 Information Security Definition

Information security is a concept that still lacks of unambiguous definitions. After many years of debates, information security has not found a worldwide definition. In fact, neither a widely accepted information security definition nor standardized critical success factors taxonomy exists [4].

Throughout the years, people and organizations responsible for critical business assets and business survivability have found different terms to make reference to the same activity: "protecting the organization's critical assets". In some cases, it is referred as information systems security, in others as digital security, or security itself. Information security regulation initiatives such as Basel II, Sarbanes-Oxley Act, and the Companies Act have defined "the need to protect information resources and prevent unauthorized access of the organization" as network security [5]. It can also be found as security management, business security, and the list goes on.

Information security, despite the variety of terms, has historically been defined as the process of ensuring information confidentiality, integrity and availability (CIA). This traditional way to understand information security helps us to handle this abstract, dynamic and complex phenomenon in a concrete way [6]. The recognized impact that human factors have on information security, along with new organizational structures which encourage employees' self control and responsibility is only one aspect, out of many others, that must be considered when reformulating a complex definition such information security.

Information security experts have tried to extent the existing CIA definition. This extension includes four human-oriented security aspects which are responsibility,

¹ The ease with which information systems can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment [3].

² Taking measurements over time and comparing two or more measurements with predefined baselines [6].

integrity, trust and ethicality [7]. The combination of these four principles with CIA was one of the first indications that highlighted the need to find a wider and more accurate information security definition. Other authors have shared their viewpoint about information security. “The discipline responsible for protecting a company’s information assets against business risks” [8]. “The degree to which malicious harm is prevented, reduced and properly reacted to” [4]. “The contributor to strengthening the organization’s ability to adapt to new risk environments and accomplish its mission” [9]. Despite the correctness of these definitions, security professionals have to understand and agree on a worldwide definition in order to make information security to work [10].

After reviewing several definitions, one thing gets clear: information security is about technology, processes and people [11]. Therefore, we propose a definition which includes the following approaches: Information security is a well-informed sense of assurance that information risks and technical, formal and informal controls are in dynamic balance. Firstly, *well-informed sense of assurance* must be achieved because if there is not knowledge and practical assurance about the organization’s status, then information security gets very hard to accomplish [10]. Secondly, *technical, formal and informal security controls* (which are synonyms of technology, processes and people) must be implemented and managed since the absence of any of them also compromise information security [12,13,14]. Lastly, *in dynamic balance* makes reference to the fact that not only these controls must be implemented and managed, but also they must be equally and dynamically treated. In fact, it has been demonstrated how using the latest technology and having security policies and procedures worth nothing if they are not upgraded on a regular basis or if the human side of security is ignored [15]. The controls mentioned above are defined as follow:

- *Technical controls*: Hardware and software tools that restrict access to buildings, rooms, computer systems and programs in order to avoid unauthorized access or incorrect uses (antivirus, firewalls, IDS, backups, etc).
- *Formal controls*: Set of policies and procedures to establish and ensure effective use of technical controls. For example, identifying roles, responsibilities, implementing indicators and training employees.
- *Informal controls*: Interventions related to deploying digital information security through the workforce by enhancing users’ willpower and willingness.

The proposed information security definition should be seen as a first approach into motivating organizations, security experts and researchers to agree upon a unique definition. A worldwide definition could help us explore new perspectives and moving from technical approaches to more business, human and law oriented definitions. We have reached a time where engineers, economist, lawyers, and policymakers must try to forge common approaches in the name of information security management improvements [16].

3 SMIS Critical Success Factors

The following critical success factors (CSFs) for SMIS implementation and maintainability have been grouped based on already defined concepts:

- The CSFs have been grouped utilizing the technical, formal and informal approach discussed above. As it was stated in the definition, these three components work together as a whole and must be equally managed.
- As a result, they have been arranged based on the “Swiss cheese” model developed by James Reason where the holes on the cheese represent equipment failures, policy’s failures or human errors waiting to happen. According to his model, the cheese has several slices (defense in depth) and each slide on the cheese represents an obstacle or defense to protect the system. When the holes in the cheese line up “trajectory”, vulnerabilities are exposed and an incident occurs [17].

However, Reason’s approach was modified, from a multi-layer approach to a 3 dimensional cheese approach. Instead, of looking at the cheese slices as layers of security, each dimension of the cheese is seen as a security control. This 3D perspective highlights how by neglecting any of the three security controls, a failure could occur. Besides, the holes are constantly moving in all the three dimensions and if there is a trajectory in any of the three axes (neglecting a single control) an unwarranted event could happen [15].

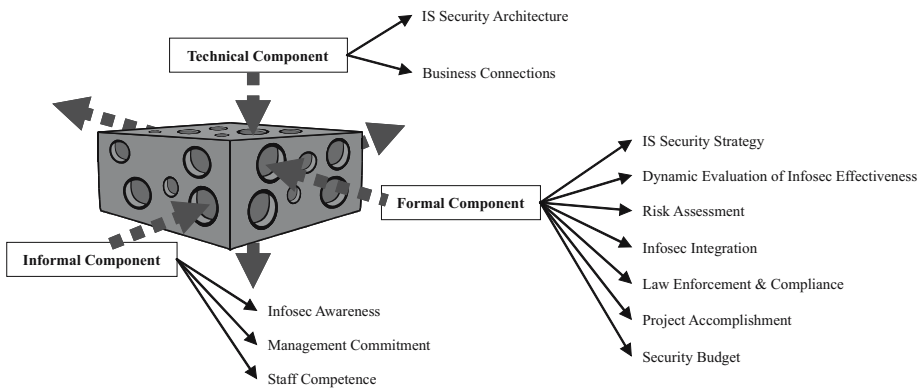


Fig. 1. CSFs arrangement using a 3D version of Reason’s Swiss cheese model

The lack of information security data and empirical studies has forced us to identify these 12 CSFs based on revising published security experts’ perspectives. However, the 76 indicators have been designed by combining the information security literature review with the data from a current project that links university researchers, IS security experts and an engineering firm which aims to measure security effectiveness.

3.1 Technical Components

3.1.1 IS Security Architecture

Defined as the way in which existent hardware and software business structures are introduced, arranged, protected and used.

Information security as well as IS reliability begins with an appropriate and robust IS security architecture design. The infrastructure that holds IS must be physically robust (indicators 1,2) and logically robust (indicators 3,4,5,6). Robust IS security architectures are also characterized by being highly dynamic (indicators 7,8) and organized (indicators 9, 10).

Table 1. IS Security Architecture

	Indicator	Formula	Unit
1	% of securized areas	$(\Sigma \text{ of securized areas} * 100) / (\text{Total areas defined as secure})$	[%]
2	% of critical equipment with adequate physical protection	$(\text{Physically protected equipment} * 100) / (\text{Critical business equipment})$	[%]
3	% of secured configurations ³	$(\Sigma \text{ of successfully secured configurations} * 100) / (\text{total configurations})$	[%]
4	Σ of users with administrator passwords per workstation	Σ of users with administrator privileges to critical workstation(s)	[users/critical workstation (s)]
5	% of users with superuser privileges	$(\Sigma \text{ of users with superuser privileges} * 100) / (\Sigma \text{ of users})$	[%]
6	% of viruses and worms hits	$(\Sigma \text{ of hits} * 100) / (\Sigma \text{ of incoming viruses per year})$	[%]
7	Σ of architecture changes	Σ of all per year	[changes/year]
8	Σ of technical internal/external audits	Σ of all per year	[audits/year]
9	% of software and hardware classified ⁴	$(\Sigma \text{ of software \& hardware classified} * 100) / (\Sigma \text{ of total software \& hardware})$	[%]
10	Σ of contracts with third parties service suppliers	$(\Sigma \text{ of contracts} * 100) / (\text{Total externalized services})$	[%]

3.1.2 Business Connections

Defined as external and internal connections to the organization's intranet or critical data.

Organizations have been approaching IS security as an external issue focusing on intrusions and connections from the outside. However, recent successful insider attacks experiences are changing the scenery making information security an internal as well as external business issue [18,19]. Security is not longer the perimeter [20] and therefore, it is vital to control remote accesses, business wireless equipment and their current security level (indicators 11,12).

The new way of doing business allows outsiders (users, clients, guests, suppliers, stakeholders, etc) access the organization's intranet or critical data. These connections represent a great advantage for organizations production-wise. However, it only takes an organization's guest connecting his/her laptop to the system, to lower your IS security level to a highly vulnerable state.

Unauthorized access testing and penetration assessments are countermeasures that should be implemented to ensure proper organizational information security from the technical point of view (indicator 13). If these connections are left unattended, then the business' IS security level will be as good as the security level of the connected device of the outsider or insider [21].

³ Activating security features by default and ensuring robust security restrictions (i.e. robust firewall configuration, Wi-Fi restrictions, etc).

⁴ To detect illegal downloaded software, verified licenses and control (if possible) external hardware devices (pendrives).

Table 2. Business Connections

	Indicator	Formula	Unit
11	Σ of remote accesses and wireless devices	Σ of accesses per month	[accesses/month]
12	Average wireless devices upgrade date (laptops)	$\frac{\Sigma (\text{date of current upgrade (s)} - \text{date of last review (s)})}{\Sigma (\text{of total wireless devices})}$	[days]
13	% of vulnerability and penetration assessments conducted	$\frac{(\Sigma \text{ of assessments conducted} * 100)}{(\Sigma \text{ of assessment scheduled})}$	[%]

3.2 Formal Components

3.2.1 IS Security Strategy

Defined as well-planned and structured SMIS improvement process. It includes having “clearly defined” SMIS plan of action’s goals, scope, resources, implementation team, their responsibilities and realistic completion times for goals set.

During the last decade, information security experts have provided organizations with several strategies to secure IS. Although they all try to achieve IS security, nearly all security experts’ strategies differ from each other. Some security experts recommend separating information security from information technology (IT) and integrate it with physical security [22]. Others highlight how security strategies based on deterrence, prevention, detection and response can be the difference between SMIS success and failure [23,24]. In addition, the information security ISO17799 recommends designing the security strategy aligned with business goals.

In order to achieve well-planned and structured security strategies, organizations need to compare their strategy with a universally accepted standard (indicator 14). Next, the strategy adopted must satisfy the organization’s security needs (indicators

Table 3. IS Security Strategy

	Indicator	Formula	Unit
14	% of strategy robustness ⁵	$\frac{(\Sigma \text{ of security actions achieved} * 100)}{(\Sigma \text{ of total actions recommend by an standard})}$	[%]
15	% of outsourced infosec processes	$\frac{(\Sigma \text{ of infosec processes outsourced} * 100)}{(\Sigma \text{ of total business processes})}$	[%]
16	Σ of audits to the infosec outsourced firm(s)	Σ of audits per year	[audits/year]
17	% of qualified IS staff ⁶	$\frac{(\text{Qualified staff} * 100)}{(\text{Average IS staff during 1 year})}$	[%]
18	Responsibility sharing ⁷	$\frac{(\Sigma \text{ of responsibilities assigned to a single staff member} * 100)}{(\Sigma \text{ of total security responsibilities})}$	[%]
19	Project delays ⁸	$\frac{((\text{Completion hours} - \text{Estimated hours for phase "n"}))}{\text{project (s)}}$	[hours behind schedule/project (s)]
20	Evolution of infosec plan of action	$\frac{(\Sigma \text{ of infosec activities from last year}) - (\Sigma \text{ of infosec activities from current year})}{\text{}}$	[infosec activities]

⁵ Using a universally accepted standard such as ISO 17799 or CobiT (If certified = 100%).

⁶ Qualified person with one or more information security certificates.

⁷ In order to achieve responsibilities assignment balance (detecting or avoiding workloads).

⁸ The project should be divided in “n” phases in order to accomplish smaller goals and witness progress.

15, 16). For example, depending on the business activity (financial vs. manufacturing) the needed strategy will vary (preventive, reactive or outsourcing). In fact, there are some information security studies that demonstrate how adopting a preventive security strategy is not the same as a corrective security strategy [25].

Proper allocation of information security human resources is the key to achieve robust information security strategies in the estimated completion time. Therefore, after evaluating the best strategy fit, organizations need to set and enforce realistic deadlines and have available qualified staff (indicators 17,18,19,20).

3.2.2 Dynamic Evaluation of Information Security Effectiveness

Defined as continuous evaluation of the SMIS’ effectiveness: Understanding and managing the highly dynamic mechanisms that control information security behaviors.

Table 4. Dynamic Evaluation of Information Security Effectiveness

	Indicator	Formula	Unit
21	Σ of internal and external systems audits	(Σ of audits per month)	[audits/month]
22	% of daily monitored processes	(Σ of infosec processes monitored daily*100)/(Σ of total business processes)	[%]
23	% of in house specialized staff dedicated to assessment of infosec activities	(Σ of hours dedicated to evaluate monthly)/(Average available qualified staff during 1 month)	[%]
24	Average time to respond to incidents	Σ (detection time (s)- response time (s) in days)/(Σ of total incidents)	[time/detected incident]
25	% of incidents stopped per month	(Σ of incidents stopped*100)/(Σ of total incidents detected per month)	[%]
26	Σ of monthly incident responses	(Σ of incidents responses*100)/(Σ of total incidents detected per month)	[%]
27	% of monthly systems' performance and assurance scheduled activities	(Σ of fixed IS anomalies*100)/(Σ of total detected anomalies per month)	[%]
28	% of nonconformity aspects fixed (found during audits activities)	(Σ of nonconformity aspects fixed*100)/(Σ of total nonconformity aspects detected)	[%]
29	% of maintenance processes executed	(Σ of maintenance process executed*100)/(Σ of total maintenance processes scheduled)	[%]
30	Σ of data recovery testing activities	(Σ of business data recovery testing activities)	[activities]

Despite the powerful high-tech security countermeasures and the available process and procedure guidelines, organizations still fail to identify the real mechanisms that control information security behaviors. The need to analyze the dynamic aspects of information security is now in evidence and some studies have already shown results [18,19,26].

Any organization using a modern operating system has to actively manage this complex and dynamically changing environment (indicators 21,22). Information security has become a challenging system to manage and near impossible to predict since attacking-tools’ developers, hackers, crackers and insiders dictate the speed of IS’ insecurity. Dynamic evaluation of information security has become the only way to keep up with the pace at which threat sophistication is traveling (indicators 23,24,25,26,27,28,29,30).

3.2.3 Risk Assessment Process

Defined as accurate identification, classification and prioritization of critical assets, vulnerabilities, threats, their impacts, and probability of happening.

Information security studies have shown that accurate risk assessment, data analysis and economic evaluations are hard to achieve because organizations find difficult to collect or keep track of information security indicators [27]. As a result, it becomes hard to correlate formal security structures and actual security behaviors.

Information security consultants and auditors have identified poor risk assessment processes as one of the most frequent reasons in SMIS implementation projects' failures [28]. On the one hand, inaccurate risk assessment processes happen when organizations do not collect and analyze information security measurements (indicators 31,32,33,34). On the other hand, when the personnel involved in the process fail to identify critical assets, areas and threats (indicators 35,36,37,38,39) due to lack of knowledge about the organization's security needs.

Table 5. Risk Assessment Process

	Indicator	Formula	Unit
31	% of Risk Assessment (R.A) automatization	$(\Sigma \text{ of automated R.A. tasks} * 100) / (\Sigma \text{ of total R.A. tasks})$	[%]
32	Current level of risk by area ⁹	Depends on the tools and methodology used	[risk level]
33	% of countermeasures implemented	$(\Sigma \text{ of implemented countermeasures} * 100) / (\Sigma \text{ of identified countermeasures})$	[%]
34	Average risk assessment review time	$(\text{Time between consecutive reviews}) / (\Sigma \text{ of reviews})$	[days]
35	Σ of high-impact incidents on processes not contemplated in previous R.A.	Σ of incidents not contemplated	[incidents not contemplated]
36	Σ of critical assets	Σ of critical assets	[assets]
37	Σ of critical areas	Σ of critical areas	[areas]
38	Σ of identified potential threats	Σ of potential threats	[threats]
39	Σ of new threats identified	$(\Sigma \text{ of threats identified during previous revision}) - (\Sigma \text{ of threats identified during current revision})$	[threats]

3.2.4 Information Security Integration

Defined as the connection between information security and the organizations' core activities and processes with the purpose of aligning information security with business objectives.

The design of IS has been focused in enhancing organizations' core competencies. Thus, it needs to be managed and secured like all the other critical business systems (indicator 40). SMIS relies on the interface between technology, policies-procedures and users. *“If you think technology can solve your security problems, then you do not understand the problems and you do not understand the technology”* [29]. Therefore, aligning end-users' information security tasks with their professional goals is a reasonable solution for such problem (indicators 41,42).

Technology has allowed us to enjoy acceptable secure IT systems. However, if end-users ignore information security activities, the system can be left susceptible to breaches and failures (indicator 43). End-users are more likely to adopt guidelines and

⁹ Applications, operating systems, servers, etc.

procedures during the SMIS implementation process if information security activities contribute to fulfilling their daily duties more effectively. The suggested indicators allow organizations to evaluate the degree of alignment between protecting the organization's core assets and processes and business objectives.

Table 6. Information Security Integration

	Indicator	Formula	Unit
40	% of systems availability	$(\Sigma \text{ of hours available} * 100) / (\text{hours expected to be available})$	[%]
41	Σ of BSP incentives	$(\Sigma \text{ of best security practice incentives given} / \text{month})$	[incentives/month]
42	Σ of protected files	$(\Sigma \text{ of files in the backup folder} * 100) / (\Sigma \text{ of critical files})$	[%]
43	Σ of point solutions (patch, access controls, etc)	$(\Sigma \text{ of infosec point solutions})$	[point solutions]

3.2.5 Project Accomplishment

Defined as the degree on which starting information security strategic, operational and technical goals are met and enforced.

Several organizations have strictly followed proper SMIS implementation methodologies but have failed to protect their critical assets. The main cause of such failure is because the level of success, implementation-wise, at which the information security goals are reached, does not meet the high level of expectations set by management or consultants. In fact, it has been demonstrated how achieving fully accomplishment of previously stated SMIS implementation goals rarely happens [2].

A qualified, involved and motivated SMIS implementation team can be the difference between SMIS implementation's failure and success (indicators 44,45,46,47). The leader of the implementation team should know the organization and possess legal knowledge (indicators 48,49). In addition, abilities such as leadership, accuracy when

Table 7. Project Accomplishment

	Indicator	Formula	Unit
44	% of policies and procedures into the design phase (not approved)	$(\Sigma \text{ of policies \& procedures into design} * 100) / (\Sigma \text{ of total p\&p identified in a standard})$	[%]
45	% of policies and procedures documented and approved	$(\Sigma \text{ of policies \& procedures approved} * 100) / (\Sigma \text{ of total p\&p identified in a standard})$	[%]
46	(average) hours dedicated to policies and procedures design	$(\text{Hours dedicated to design}) / (\text{Average qualified staff during 1 year})$	[hours/team member-year]
47	(average) hours dedicated to policies and procedures implementation	$(\text{Hours dedicated to implement}) / (\text{Average qualified staff during 1 year})$	[hours/team member-year]
48	(average) hours dedicated to policies and procedures reviews and upgrading activities	$(\text{Hours dedicated to review and upgrade}) / (\text{Average qualified staff during 1 year})$	[hours/team member-year]
49	Internal audits	$(\Sigma \text{ of internal audits perform} * 100) / (\Sigma \text{ of internal audits scheduled or planned})$	[%]
50	Maturity level of current controls	$(\Sigma \text{ of incidents responses from current year}) - (\Sigma \text{ of incidents responses from previous year})$	[incidents responses]

estimating project costs, realistic evaluation of the “before and after” information security situation, downstream-upstream communication and time availability for information security activities, are key factors for successful SMIS implementation and maintainability (indicator 50).

3.2.6 Law Enforcement and Compliance

Defined as the degree of enforcement and compliance of implemented information security controls. Externally done by regulatory institutions such as the Sarbanes Oxley Act and the Spanish LOPD¹⁰ and internally done through internal controls within the organization.

New regulations such as the Sarbanes-Oxley Act and the Spanish LOPD have had a visible impact on SMIS implementations. These regulations are slowly forcing organizations to increase their information security level. However, some organizations have taken these regulations as their security strategy by default. Adopting regulations as security strategy can lead organizations towards the compliance requirements of the moment leaving critical assets unattended. Regulations not only change security focus but also can cause costly and inefficient investments when organizations only consider assets and processes subjected to regulations [9].

At first, organizations should start by separating the operational part of information security from the compliance and enforcing part [30] in order to execute effective internal audits (indicator 51). Next, “no-compliance” severe sanctions should be applied to disobey departments within the organization (indicator 52). If so, they would not have a choice but to obey the information security practices implemented. Higher degrees of enforcement as well as no-compliance severe sanctions increase deterrence mechanisms that in the long run prevent incidents from happening [23].

Table 8. Law Enforcement and Compliance

	Indicator	Formula	Unit
51	% of fulfilled regulations ¹¹	$(\Sigma \text{ of regulations fulfilled} * 100) / (\Sigma \text{ of regulations enforced by authorities})$	[%]
52	% of penalties imposed to users ¹²	$(\Sigma \text{ of penalties imposed} * 100) / (\Sigma \text{ of infosec bad practices detected})$	[%]

3.2.7 Budget

Defined as the percentage of IT economical resources dedicated to information security.

Organizations do not routinely require return of investment calculations on security investments since so far, information security activities have being treated as expenses [31]. Operational expenses (patch a breach) are usually easy to justify budget-wise but information security capital investments are not straightforward. As information security gets more expensive, infosec leaders are asked to show infosec budget allocations as well as cost-benefits analysis (indicators 53,54).

¹⁰ Ley Orgánica de Protección de Datos www.belt.es/legislacion/

¹¹ Confidential data only for organization internal purposes .

¹² Confidential data only for organization internal purposes.

The lack of information security measurements and analysis makes the *IT vs. security* investments grow at disproportional rates. If the IT budget grows in isolation, then the budget needed to keep the system in a reliable state becomes highly expensive. Therefore, organizations need to track the evolution of both budgets (indicator 55). They should also perceive security expenses as an opportunity to improve upon IS availability, reputation and ability to accomplish business mission and ability to adapt to changing risk environments [9].

Table 9. Budget

	Indicator	Formula	Unit
53	Security budget segregation ¹³	$(\text{Budget spent on the analyzed area} * 100) / (\text{Security budget})$	[%]
54	Σ of cost-benefit analysis (NPV, IRR, ROSI, ALE, GLEIS) ¹⁴	$(\Sigma \text{ of cost-benefit analysis})$	[analysis/year]
55	Security budget evolution	$(\text{Budget spent on IS security} * 100) / (\text{IT budget every year})$	[%]

3.3 Informal Components

3.3.1 Information Security Awareness

Defined as the appreciation, at all levels within the organization, about the needs and benefits of information security.

Organizations have misplaced information security resources in the IT department and failed to identify people’s operational weaknesses as the root causes of the majority of information security breaches [9]. Information security must be much more than legislators, policies, and procedures [32]. It is a business problem and a people’s problem that requires active, involve, and aware users [8].

Information security awareness will receive more attention this year than any other because current insider threat are called to be the biggest information security threat during 2006. This upcoming year, attackers will not spend time looking for system’s vulnerabilities. Instead, they will focus on convincing employees to execute cyber attacks [33]. Therefore, developing information security awareness among the organization is no longer a choice but a necessity [34].

Security awareness is one of the few countermeasures capable to stop security incidents motivated by greed, economic and other personal problems [13]. By improving end-user behavior, organizations can minimize insider threat probability as well as approach security more as a business problem instead of a merely technical issue (indicators 56,57,58).

In the near future, role model organizations are going to be characterized by promoting ongoing users’ training and education programs in information security as well as effective communication of information security goals (indicator 59). They will also achieve robust SMIS (indicators 60,61) and fast incident responses (indicators 62,63).

¹³ Critical assets and critical areas first.

¹⁴ To compare the operational cost of security vs. the investment cost.

Table 10. Information Security Awareness

	Indicator	Formula	Unit
56	Incidents reported per employee	$(\Sigma \text{ of incidents reported})/(\text{Average employees during 1 year})$	[incidents/user-year]
57	Average training hours received per year	$(\text{Hours dedicated to infosec training})/(\text{Average staff during 1 year})$	[hours/user-year]
58	Degree of awareness (by type of user: management, IT staff, end users, etc) ¹⁵	$(\text{Survey score} * 100) / (\text{Optimal survey score})$	[%]
59	Σ of infosec-related reports, newsletters, memorandums sent	$(\Sigma \text{ of memorandums sent per year})$	[memorandum/year]
60	% of incidents investigated	$(\Sigma \text{ of incidents investigated} * 100) / (\Sigma \text{ of detected incidents per year})$	[%]
61	Certification status ¹⁶	$(\Sigma \text{ of hours needed to achieve certification})$	[hours]
62	Business critical data recovery time ¹⁷	$\Sigma \text{ of hours needed to recover lost critical data or system functioning}$	[hours]
63	Average critical data recording date ¹⁸	$(\text{Time between consecutive recordings})$	[days or hours]

3.3.2 Information Security Awareness

Defined as the degree of understanding and support from top management about the impact of information security on the business future and stakeholders.

Organizations’ dependency on IS, information security legal regulations and business competition are triggering top management commitment. Their support and understanding are critical because business decisions usually drive technical and operational decisions (indicators 64,65,66).

Table 11. Information Security Awareness

	Indicator	Formula	Unit
64	% of SMIS policies approved	$(\Sigma \text{ of policies approved} * 100) / (\Sigma \text{ of policies suggested per year})$	[%]
65	% of SMIS procedures approved	$(\Sigma \text{ of procedures approved} * 100) / (\Sigma \text{ of procedures suggested per year})$	[%]
66	% of security budget spent on training	$(\text{Budget spent on training} * 100) / (\text{Security budget})$	[%]
67	Downstream infosec communication	$(\Sigma \text{ of meeting between infosec leaders \& top management} * 100) / (\Sigma \text{ of meetings between department leaders \& top management per year})$	[%]
68	Average training hours received per year	$(\text{Hours dedicated to training}) / (\text{Average staff during 1 year})$	[hours/user-year]
69	% of satisfactory accomplishment per training activity	$(\text{Average of satisfaction} * 100) / (\Sigma \text{ of training activities per year})$	[%]
70	% of infosec reports asked	$(\Sigma \text{ of infosec reports asked} * 100) / (\Sigma \text{ of business reports asked per year})$	[%]

¹⁵ Done through an internal information security survey.

¹⁶ The accuracy of this indicator is no the final aim but to situate the business with respect to others. Measured in hours or also in requirements needed to achieve certification.

¹⁷ The higher the IS dependency, the shorter the time.

¹⁸ The higher the IS dependency, the shorter the time.

Underestimating information security is not the only critical factor. The degree of information security top-management commitment varies across organizations' size. Top management from large corporations has been practically forced to commit to information security due to companies' fusions, external regulations and international business relations. However, in small and medium size enterprises (SMEs), management's involvement tends to be much lower since they do not see themselves as potential targets for cyber attackers [27].

In order to raise top-management commitment, regardless of enterprise size, information security must be directly related to business success. If the responsible for SMIS in the organization makes top-management understand that information security is the discipline which mitigates business risks, then effective security decisions will be made (indicators 67,68,69,70).

3.3.3 Administrators and End Users Competence

Defined as IT knowledge and skills that can be used to properly utilize and secure IS but also used toward exploiting dishonest advantages through the use of IS.

Computing practices of system administrators and users continue to be one of the greatest information security challenges. People who administer IT systems sometimes practice insecure practices "shortcuts" claiming IS efficiency's improvements or simply being helping certain end users. For example, a system administrator who changes a firewall rule, despite security rules and management approval, in order to help a remote user who has trouble sending email [35].

Between 80 and 90 % of organizational problems are due to human errors or bad practices [17]. Therefore, having honest, competent, smart and skillful systems administrators is a CSF for ensuring SMIS (indicators 71,72,73,74). *It does not matter how IS are protected or the safety devices that have been settled, what is really important is who is using and defending the system* [36].

End-users competence though, has the exact opposite effect on information security. Recent end user sophistication studies have demonstrated the strong

Table 12. Administrators and End Users Competence

	Indicator	Formula	Unit
71	Staff responsible for infosec training hours received per year ¹⁹	(Hours dedicated to training)/(Average infosec staff during 1 year)	[hours/team member-year]
72	% of qualified IS staff ²⁰	(Qualified staff*100)/(Average IS staff during 1 year)	[%]
73	% of satisfactory accomplishment per training activity	(Average of satisfaction*100)/(Σ of training activities per year)	[%]
74	Upstream infosec communication	(Σ of meetings or reports with executive management per year)	[meetings/year]
75	% of reported incidents in users' PCs	(Σ of reported incidents in users' PCs*100)/(Σ of total reported incidents per year)	[%]
76	Degree of organizational climate satisfaction ²¹	(Survey score*100)/(Optimal survey score)	[%]

¹⁹ Due to threat sophistication and technology dynamism.

²⁰ Person with one or more information security certificate.

²¹ Paying close attention to users' behavior or attitude can be a powerful insider threat indicator.

relationship between end user sophistication and the potential of misusing IT systems [37]. These studies can be corroborated by the fact that, the majority of computer abuse comes from current employees who have managed to modify or swerve existing security controls [14]. Therefore, segregation of duties, enhancing incident reporting and monitoring users' behaviors are effective countermeasures to protect business critical assets and areas (indicator 75,76).

4 Conclusions

The presented SMIS critical success factors and indicators are the result of a combination of current information security literature, security experts' perspectives and an ongoing project which is trying to identify and simulate the security structure that generates current organizational security behaviors. The current standing information security situation indicates how the actual focus of information security research is not coinciding with the most critical factors and even worse there are not well-defined methods to measure information security.

Identifying information security CSFs and measuring the countermeasures' effectiveness is a common goal for almost all organizations today. These two critical activities are, to a great extent, an internal activity for organizations. Therefore, as information systems become more complex and indispensables, getting feedback and measuring the level of information systems performance represents more than ever a business priority [6].

This work has allowed us to corroborate that current organizations struggle with the most basic but still unknown information security interrogates. How much security does my organization need? How much do we currently have? What to measure? And probably the most important question: How should we measure it? These questions must be properly answered in order to successfully implement, maintain and manage the security of information systems.

We predict that the future of information security will be based on the dynamic balance between technology (technical), processes (formal) and people (informal). The majority of the reviewed authors coincide in the fact that these 12 CSFs are going to be crucial to improve the overall organization's critical assets protection. However, it is important to highlight that organizations should only choose the CSFs that better fit their security needs and implement the associated indicators. Although, these 12 CSFs are the most demanded by current information security practitioners, it is not strictly necessary to manage and implement all of them.

Achieving bulletproof information security is simply impossible or just too expensive. However, by implementing and analyzing these simple but helpful set of indicators, organizations will be able to easily witness their SMIS project' evolution as well as measure its effectiveness. Only after that, security resources will be allocated and used more efficiently and accurate assets and economic evaluations will be achieved.

References

1. Ernst&Young.: Global Information Security Survey (2002) www.ey.com
2. Bjorck, F.: Institutional Theory: A New Perspective for Research into IS/IT Security in Organizations. Proceedings of the 37th Hawaii International Conference on System Sciences. (2004)
3. Institute of Electrical and Electronics Engineers: IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY (1990)
4. Firesmith, D.G.: Common Concepts Underlying Safety, Security and Survivability Engineering. December (2003) CMU/SEI-2003-TN-033.
5. Burling, M.: The key to compliance. www.net-security.org.
6. Kajava, J., Savola, R.: Towards Better Information Security Management by Understanding Security Metrics and Measuring Processes (2005)
7. Dhillon, G., Backhouse, J.: Information System Security Management in the New Millennium. Communication of the ACM. July (2000) Vol. 43. No. 7
8. Von Sloms, S.H., Von Sloms, R.: From Information Security to.... Business Security? Computer & Security (2005) Vol. 24 271-273
9. Caralli, R. A., Wilson, W. R.: The challenges of Security Management. Networked Systems Survivability Program, SEI.
10. Anderson, James. M.: Why We Need a New Definition of Information Security.
11. Schneier, B.: Monthly Newsletter www.schneier.com
12. Dhillon, G.: Managing and Controlling Computer Misuse. Information Management & Computer Security. 7/4 (1999) 171-175.
13. Dhillon, G.: Violating of Safeguards by Trusted Personal and Understanding Related Information Security Concerns. Computer & Security Vol 20 No. 2 (2001) 165-172
14. Dhillon, G., Moores, S.: Computer crimes: Theorizing About the Enemy Within. Computer & Security Vol 20 No. 8 (2001) 715-723
15. Torres, J. M. Sarriegui, J. M.: Dynamics Aspects of Security Management of Information Systems. Proceedings of Systems Dynamic Society Conference, July (2003) Oxford, UK.
16. Anderson, R.: Why Information Security is Hard: An Economic Perspective (2001)
17. Reason, J.: Managing the Risk of Organizational Accidents. Hants, UK: Ashgate Publishing Ltd, (1997)
18. Andersen, D., Cappelli, D., Gonzalez, J., Mojtahedzadeh, M., Moore, A., Rich, E., Sarriegui, J.M., Shimeall, T., Stanton, J., Weaver, E., Zagonel, A.: Preliminary System Dynamics Maps of the Insider Cyber-Threat Problem. Proceedings of System Dynamics Society Conference. Oxford, UK (2004)
19. Melara, C., Sarriegui, J.M., Gonzalez, J., Sawicka, A., Cooke, D.L.: A System Dynamics Model of an Insider Attack on an Information System. In: From Modeling to Managing Security: A System Dynamics Approach, Norwegian Academic Press Kristians, Norway 2003
20. Wilson, S.: The Future of Vulnerability Management: Information Security Bulletin, Vol 8 March (2003) 69.
21. Schneier, B.: Information Security Management. Conference in Bilbao, Spain (2005)
22. Berinato, S., Cosgrove, L.: Six Secrets of Highly Secure Organizations. CIO magazine, Sep. 15 (2004)
23. Theoharidou, M., Karida, M., Kokolakis, S.: The Insider Threat to Information Systems and the Effectiveness of ISO 17799. Computer&Security 24 (2005) 472-848
24. Parker, D.: Fighting Computer Crime. New York, NY. John Wiley & Sons (1998)

25. Torres. J. M., Sarriegui J. M., Santos. J.: Searching for Preventive-Corrective Security Balance. Proceedings of Systems Dynamic Society Conference, Boston U.S.A, July (2005)
26. Gonzalez. J., Rich. E.: Helping Prevent Information Security Risks in the Transition to Integrated Operations. *Teletronikk* 1, (2005)
27. Sarriegui. J. M., Eceiza. E., Torres. J. M. Santos. J.: Security Management of Information Systems Report (2005)
28. Bjorck. F.: Implementing Information Security Management System: Empirical Study of Critical Success Factors.
29. Schneier. B.: *Applied Cryptography: Protocols, Algorithms and Source Code* in C. New York: John Wiley& Sons, Inc., (1994)
30. Von Sloms. S.H.: Information Security Governance: compliance management vs. operational management. *Computer&Security* 24 (2005) 433-447
31. Gordon. L., Loeb. M.: *Managing Cyber Security Resources. A cost-benefit analysis.* New York, NY: McGraw-Hill (2006)
32. Schneier. B.: *Beyond Fears.* 1 edn. New York, NY: Copernicus Book (2003)
33. IBM Global Business Security index survey. Potential threats to information security during 2006 (2005)
34. Mitnick. K.: *The Art of Deception.* Indianapolis, Indiana: John Wiley, Inc (2002)
35. Schultz. E.: The human Factor in Security. *Computer&Security* 24 (2005) 425-426
36. Schneier. B.: *Managed Security Monitoring: Network Security for the 21st Century.* *Computer and Security* 20 (2001) 491-503
37. Magklaras. G.B., Furnell S.M.: A Preliminary Model of End User Sophistication for Insider Threat Prediction in IT Systems. *Computer&Security* 24 (2005) 371-380

Author Index

- Ackley, Elena S. 72
Almeroth, Kevin 343
Anagnostakis, Kostas 427
Assora, Mohammed 489
Atzori, Maurizio 60
- Baek, Joonsang 217
Bagga, Walid 233
Banks, Greg 343
Berta, István Zsolt 246
Bononi, Luciano 398
Burnside, Matthew 32
Byun, Yung-Cheol 415
- Chen, Kefei 359
Chou, Chun-Yen 161
Cova, Marco 343
Cuellar, Jorge 502
- Danezis, George 46
De Cannière, Christophe 171
de Vink, E.P. 476
de Weger, Benne 203
Deng, Robert H. 359
Desmedt, Yvo 459
Dunkelman, Orr 85
- Esponda, Fernando 72
- Felmetsger, Viktoria 343
Forrest, Stephanie 72
- Greenwald, Michael B. 427
Guo, Jie 359
- Helman, Paul 72
Herranz, Javier 117
Hu, Yuh-Hua 161
Huber, Ulrich 270
- Imai, Hideki 443
Ioannidis, Sotiris 299, 427
- Jeong, Taikyeong T. 415
Jia, Haixia 72
- Jochemsz, Ellen 203
Jonker, H.L. 476
- Kang, Jin-Suk 415
Kanti Das, Tanmoy 257
Katzarova, Mila 517
Kawamura, Shinichi 146
Keller, Nathan 85
Kemmerer, Richard 343
Keromytis, Angelos D. 32, 427
Kim, Jongsung 85
Kirda, Engin 1
Klinkoff, Patrick 1
Kobara, Kazukuni 443
Komano, Yuichi 146
Kotenko, Igor 327
Kruegel, Christopher 1
- Laguillaumie, Fabien 117
Lai, Feipei 161
Lee, Won-Suk 415
Li, Guole 17
Li, Jun 373
Li, Yingjiu 359
Lin, Zhiqiang 17
Lu, Jiqiang 85
- Mao, Bing 17
Marques, Hugo 311
Mendel, Florian 101
Molva, Refik 233
- Oh, Sang-Hyun 415
Oheimb, David von 502
Ohta, Kazuo 146
- Pieprzyk, Josef 459
Pramstaller, Norbert 101
Prevelakis, Vassilis 299
- Rechberger, Christian 101
Ren, Michał 257
Rijmen, Vincent 101

- Sadeghi, Ahmad-Reza 270
Safavi-Naini, Reihaneh 131, 217
Santos, Javier 530
Sarriegi, Jose M. 530
Serrano, Nicolás 530
Shaked, Yaniv 187
Shimbo, Atsushi 146
Shin, SeongHan 443
Shirvani, Ayoub 489
Simpson, Andrew 517
Steinfeld, Ron 459
Stephen Huang, Shou-Hsuan 383
Stütz, Thomas 286
Susilo, Willy 131, 217
- Tacconi, Carlo 398
Tonien, Dongvu 131
Torres, Jose M. 530
- Uhl, Andreas 286
Ulanov, Alexander 327
- Vigna, Giovanni 1, 343
- Wang, Huaxiong 459
Wang, Lih-Chung 161
Wool, Avishai 187
- Xia, Nai 17
Xie, Li 17
- Yang, Jianhua 383
Yin, Huifang 373
- Zhang, Yongzhong 383
Zhou, Jianying 257
Zúquete, André 311