

Steven M. Bellovin
Rosario Gennaro
Angelos Keromytis
Moti Yung (Eds.)

LNCS 5037

Applied Cryptography and Network Security

6th International Conference, ACNS 2008
New York, NY, USA, June 2008
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

Alfred Kobsa

University of California, Irvine, CA, 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

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Steven M. Bellovin Rosario Gennaro
Angelos Keromytis Moti Yung (Eds.)

Applied Cryptography and Network Security

6th International Conference, ACNS 2008
New York, NY, USA, June 3-6, 2008
Proceedings

Volume Editors

Steven M. Bellovin
Angelos Keromytis
Columbia University
1214 Amsterdam Avenue, New York, NY 10027, USA
E-mail: {smb,angelos}@cs.columbia.edu

Rosario Gennaro
IBM T.J.Watson Research Center
19 Skyline Dr., Hawthorne, NY 10532, USA
E-mail: rosario@us.ibm.com

Moti Yung
Google Inc.
and
Columbia University, Department of Computer Science
1214 Amsterdam Avenue, New York, NY 10027, USA
E-mail: moti@cs.columbia.edu

Library of Congress Control Number: 2008927608

CR Subject Classification (1998): E.3, C.2, D.4.6, H.4, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-68913-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-68913-3 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 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12278923 06/3180 5 4 3 2 1 0

Preface

ACNS 2008, the 6th International Conference on Applied Cryptography and Network Security, was held in New York, New York, June 3–6, 2008, at Columbia University. ACNS 2008 was organized in cooperation with the International Association for Cryptologic Research (IACR) and the Department of Computer Science at Columbia University. The General Chairs of the conference were Angelos Keromytis and Moti Yung.

The conference received 131 submissions, of which the Program Committee, chaired by Steven Bellovin and Rosario Gennaro, selected 30 for presentation at the conference. The Best Student Paper Award was given to Liang Xie and Hui Song for their paper “On the Effectiveness of Internal Patch Dissemination Against File-Sharing Worms” (co-authored with Sencun Zhu).

These proceedings consist of revised versions of the presented papers. The revisions were not reviewed. The authors bear full responsibility for the contents of their papers.

There were many submissions of good quality, and consequently the selection process was challenging and very competitive. Indeed, a number of good papers were not accepted due to lack of space in the program. The main considerations in selecting the program were conceptual and technical innovation and quality of presentation. As reflected in the Call for Papers, an attempt was made to solicit and publish papers suggesting novel paradigms, original directions, or non-traditional perspectives.

We would like to extend our heartfelt thanks to the Program Committee members, who dedicated so much time and effort to provide a thorough and in-depth review of the submissions, with high standards of professional integrity. We also thank the many external reviewers who assisted the Program Committee in its work. Most importantly, we thank the authors of submitted papers for their contributions; without these papers, after all, there would be no ACNS conference.

A special thanks is due to Shai Halevi for writing the software that greatly facilitated the committee work, and for his responsiveness in answering all our questions.

We are grateful to Jianying Zhou who, as Publicity Chair, relentlessly advertised the conference, to Angelika Zavou for her timely maintenance of the conference website and to Sophie Majewski for helping with the local arrangements.

Finally, we appreciate the assistance provided by the Springer LNCS editorial staff in assembling these proceedings.

June 2008

Steven Bellovin
Rosario Gennaro
Angelos Keromytis
Moti Yung

ACNS 2008

6th Annual Conference on Applied Cryptography and Network Security

Columbia University, New York, NY, USA
June 3–6, 2008

General Chairs

Angelos Keromytis, Columbia University
Moti Yung, Google Inc.

Program Chairs

Steven M. Bellovin, Columbia University
Rosario Gennaro, IBM Research

Program Committee

Masayuki Abe	NTT, Japan
Ben Adida	Harvard University, USA
Feng Bao	Institute for Infocomm Research, Singapore
Lujo Bauer	CMU, USA
Giampaolo Bella	University of Catania, Italy
Steven Bellovin	Columbia University, USA
John Black	University of Colorado, USA
Nikita Borisov	University of Illinois Urbana-Champaign, USA
Colin Boyd	Queensland University of Technology, Australia
Dario Catalano	University of Catania, Italy
Debra Cook	Alcatel-Lucent Bell Labs, USA
Alexander W. Dent	Royal Holloway, University of London, UK
Nelly Fazio	IBM Research, USA
Marc Fischlin	Darmstadt University of Technology, Germany
Debin Gao	Singapore Management University, Singapore
Rosario Gennaro	IBM Research, USA
Peter Gutmann	University of Auckland, New Zealand
John Ioannidis	Packet General Networks, USA
Stanislaw Jarecki	University of California Irvine, USA
Ari Juels	RSA Laboratories, USA
Kaoru Kurosawa	Ibaraki University, Japan
Yehuda Lindell	Bar-Ilan University, Israel

Moses Liskov	The College of William and Mary, USA
Javier Lopez	University of Malaga, Spain
Jelena Mirkovic	USC/ISI, USA
David Naccache	Ecole Normale Superieure, France
Alina Oprea	RSA Laboratories, USA
Tom Shrimpton	Portland State University, USA
Jonathan Smith	University of Pennsylvania, USA
Angelos Stavrou	George Mason University, USA
Xiaoyun Wang	Shandong University, China
Nicholas Weaver	ICSI Berkeley, USA
Steve Weis	Google, USA
Tara Whalen	Dalhousie University, Canada
Michael Wiener	Cryptographic Clarity, Canada
Avishai Wool	Tel-Aviv University, Israel
Diego Zamboni	IBM Research, Switzerland
Jianyng Zhou	Institute for Infocomm Research, Singapore

External Reviewers

Cristina Alcaraz	Swee-Huay Heng	Chris Peikert
Joonsang Baek	Shoichi Hirose	Thomas Peyrin
Daniel Bailey	Huseyin Hisil	Bart Preneel
Mira Belenkyi	Christian Hoertnagl	Mario Di Raimondo
Vicente Benjumea	Susan Hohenberger	Vincent Rijmen
Kevin Bowers	Stefan Katzenbeisser	Rodrigo Roman
Christian Cachin	Gunes Kayacik	Alex Ross
Carlos Cid	Takeshi Koshihara	Jason Rouse
Mauro Conti	Anja Lehmann	Yu Sasaki
Nicolas Courtois	Andrew Lewis	Dominique Schroeder
Erik Dahmen	Francesco Librizzi	Matthias Schunter
Ehud Doron	Jennifer Lindsay	Elizabeth C. Schwartz
Maria Fernandez-Gago	Norka Lucena	Amir Shenhav
Dario Fiore	Ling Cheung	Nigel Smart
Keith Frikken	Jean Martina	Alessandro Sorniotti
Jun Furukawa	David Molnar	Natasa Terzija
David Galindo	Steven Murdoch	Yuuki Tokunaga
Deepak Garg	Pablo Najera	Eran Tromer
Scott Garriss	Aleksandra Nenadic	Duong Quang Viet
Carrie Gates	Antonio Nicolosi	Zhaohui Wang
Chris Giblin	Wakaha Ogata	Andreas Westfeld
David Goldenberg	Katsuyuki Okeya	Juerg Wullschleger
Juan González	Dan Page	Yanjiang Yang
Choudary Gorantla	Pascal Paillier	Lei Zhang
Robbert de Haan	Kenneth G. Paterson	
Shai Halevi	Maura Paterson	

Table of Contents

On the Effectiveness of Internal Patching Against File-Sharing Worms	1
<i>Liang Xie, Hui Song, and Suncun Zhu</i>	
Peeking Through the Cloud: DNS-Based Estimation and Its Applications	21
<i>Moheeb Abu Rajab, Fabian Monrose, Andreas Terzis, and Niels Provos</i>	
Pushback for Overlay Networks: Protecting Against Malicious Insiders	39
<i>Angelos Stavrou, Michael E. Locasto, and Angelos D. Keromytis</i>	
PPAA: Peer-to-Peer Anonymous Authentication	55
<i>Patrick P. Tsang and Sean W. Smith</i>	
Generic Constructions of Stateful Public Key Encryption and Their Applications	75
<i>Joonsang Baek, Jianying Zhou, and Feng Bao</i>	
Traceable and Retrievable Identity-Based Encryption	94
<i>Man Ho Au, Qiong Huang, Joseph K. Liu, Willy Susilo, Duncan S. Wong, and Guomin Yang</i>	
Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures	111
<i>Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta</i>	
Attacking Reduced Round SHA-256	130
<i>Somitra Kumar Sanadhya and Palash Sarkar</i>	
DAKOTA – Hashing from a Combination of Modular Arithmetic and Symmetric Cryptography	144
<i>Ivan B. Damgård, Lars R. Knudsen, and Søren S. Thomsen</i>	
Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA?	156
<i>Yevgeniy Dodis and Prashant Puniya</i>	
Replay Attack in a Fair Exchange Protocol	174
<i>Macià Mut-Puigserver, Magdalena Payeras-Capellà, Josep Lluís Ferrer-Gomila, and Llorenç Huguet-Rotger</i>	
Improved Conditional E-Payments	188
<i>Marina Blanton</i>	

Anonymity in Transferable E-cash	207
<i>Sébastien Canard and Aline Gouget</i>	
Generic Security-Amplifying Methods of Ordinary Digital Signatures . . .	224
<i>Jin Li, Kwangjo Kim, Fanguo Zhang, and Duncan S. Wong</i>	
New Differential-Algebraic Attacks and Reparametrization of Rainbow	242
<i>Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng</i>	
Trapdoor Sanitizable Signatures and Their Application to Content Protection	258
<i>Sébastien Canard, Fabien Laguillaumie, and Michel Milhau</i>	
Multi-factor Authenticated Key Exchange	277
<i>David Pointcheval and Sébastien Zimmer</i>	
Repelling Detour Attack Against Onions with Re-encryption	296
<i>Marek Klonowski, Mirosław Kutylowski, and Anna Lauks</i>	
Analysis of EAP-GPSK Authentication Protocol	309
<i>John C. Mitchell, Arnab Roy, Paul Rowe, and Andre Scedrov</i>	
Efficient Device Pairing Using “Human-Comparable” Synchronized Audiovisual Patterns	328
<i>Ramnath Prasad and Nitesh Saxena</i>	
PUF-HB: A Tamper-Resilient HB Based Authentication Protocol	346
<i>Ghaith Hammouri and Berk Sunar</i>	
An Authentication Scheme Based on the Twisted Conjugacy Problem	366
<i>Vladimir Shpilrain and Alexander Ushakov</i>	
Restricted Queries over an Encrypted Index with Applications to Regulatory Compliance	373
<i>Nikita Borisov and Soumyadeb Mitra</i>	
A Practical and Efficient Tree-List Structure for Public-Key Certificate Validation	392
<i>Tong-Lee Lim, A. Lakshminarayanan, and Vira Saksen</i>	
On the Security of the CCM Encryption Mode and of a Slight Variant	411
<i>Pierre-Alain Fouque, Gwenaëlle Martinet, Frédéric Valette, and Sébastien Zimmer</i>	
$wNAF^*$, an Efficient Left-to-Right Signed Digit Recoding Algorithm . . .	429
<i>Brian King</i>	

A Very Compact “Perfectly Masked” S-Box for AES	446
<i>D. Canright and Lejla Batina</i>	
Steel, Cast Iron and Concrete: Security Engineering for Real World Wireless Sensor Networks	460
<i>Frank Stajano, Dan Cvrcek, and Matt Lewis</i>	
Traceable Privacy of Recent Provably-Secure RFID Protocols.....	479
<i>Khaled Ouafi and Raphael C.-W. Phan</i>	
The Security of EPC Gen2 Compliant RFID Protocols.....	490
<i>Mike Burmester and Breno de Medeiros</i>	
Author Index	507

On the Effectiveness of Internal Patching Against File-Sharing Worms

Liang Xie¹, Hui Song³, and Suncun Zhu^{1,2}

¹ Department of Computer Science and Engineering, The Pennsylvania State University

² College of Information Sciences and Technology, The Pennsylvania State University

³ Computer Science Department, Frostburg State University

lxie@cse.psu.edu, hsong@frostburg.edu, szhu@cse.psu.edu

Abstract. File-sharing worms have been terrorizing Peer-to-peer (P2P) systems in recent years. Existing defenses relying on users' individual recoveries or limiting users' file-sharing activities are ineffective. Automated patching tools such as Microsoft Windows Update and Symantec Security Update are currently the most popular vehicles for eliminating and containing Internet worms, but they are not necessarily the best fits for combating P2P file-sharing worms, which propagate within a relatively smaller community. In this paper, we propose a complementary P2P-tailored patching system which utilizes the existing file-sharing mechanisms to internally disseminate security patches to those participating peers in a timely and distributed fashion. Specifically, we examine the effectiveness of leveraging the file downloading or searching process to notify vulnerable end hosts of the surging worms and push corresponding security updates to these hosts. We show through in-depth analysis and extensive experiments that both methods are scalable and effective in combating existing P2P worms.

1 Introduction

P2P file-sharing programs such as KaZaA, iMesh, Morpheus are popular Internet applications that allow users to download and share electronic files. As one of the most popular networks, KaZaA has four million simultaneous users. The powerful data access feature, however, brings about unique privacy and security threats in these systems. In addition to adware and spyware, P2P hosts are placed at the risk of various viruses and malicious codes [13].

Our focus in this paper is *file-sharing worms* [1], which are malware spreading through file-sharing activities within P2P systems. Specifically, a user searches for a file in the network and acquires a list of accessible targets among which there could be a disguised one provided by some infected machine. Unwittingly she downloads the file and opens it, resulting in her own machine being infected. Recently, many file-sharing worms have been reported, e.g., Benjamin.a, Franvir, Bare.a, Darby.m, and Duload worm that are actively attacking KaZaA, Gnutella, and eDonkey2000 networks [3,7]. One experimental study reported that 44% of the 4,788 executable files downloaded through a KaZaA

¹ Like mass-mailing worms, file-sharing worms also require human's operations to propagate. As such, they are sometimes referred to as viruses or malware.

client program contain malicious code [25]; another experimental study revealed that 12% of the KaZaA client hosts were infected by over 40 different worms in February and May, 2006 [21]. Some disastrous consequences of attacks from file-sharing worms include opening backdoors, changing system registries and client configurations, and collecting clients' confidential data [7].

It would not be surprising that worms will increasingly exploit file-sharing applications as their major infection vector. However, research on these imminent threats has just started and most existing work focused on modeling worm behavior such as infections and propagation in file-sharing environments [11][17][23]. The problem of quarantining these worms has not been adequately addressed. Currently the best defense mirrors the strategy against Internet computer viruses with the inception of security patches from vendors (e.g., Microsoft, Symantec, McAfee). The method of automated security update is widely employed by security servers to automatically push the latest security patches to Internet hosts [24]. This generic solution certainly helps protect Internet hosts at the earliest stage of worm spreads, but it is not P2P-oriented. That is, security servers either have to blindly deliver P2P patches to all the Internet hosts (including those non-P2P machines and those who have installed P2P software but are not executing it), or have to scan for currently running P2P client programs within each Internet host before sending a patch to it. In both cases, system resources and network bandwidth could be greatly wasted. Moreover, unnecessary security updates could cause annoying machine reboots (sometimes required for a complete security update), which unavoidably interrupt those non-related users' on-going tasks running on their hosts.

Contribution: In this paper, we study the feasibility of utilizing the existing file-sharing infrastructure to internally push security updates to the participating nodes in P2P systems. We propose two BitTorrent-like mechanisms for distributing the security patches. In file-sharing networks such as Gnutella, a very small fraction (5%) of hosts usually provide a large fraction of the shared files (70%) [14]. Exploiting this asymmetry in file-sharing, we consider first disseminating the security patches to these popular hosts, such that most of the other participating hosts can receive the patches from these popular hosts when they actively download files from them. Our second approach is based on the belief that P2P users as a community should help each other in combating worm attacks. Therefore, when a host detects worm infection from a downloaded file, it first re-performs a search on the infected file to identify those hosts possessing the same file, and then it collaboratively notifies these hosts of the worm information as well as the security patch. Based on a modified fluid model, we analyze worm spreads and evaluate the effectiveness of our approaches in unstructured networks. Our result demonstrates that both schemes can help a file-sharing system with 20, 000 hosts achieve a high immunity rate (90%) within a few dozens of hours after the initial worm surge.

Our solutions are not a substitution for the existing automatic patching systems but rather a nice complement to them. Our proposed techniques are not necessarily very complex, but our work, backed up with solid analytic modelling and extensive experiments, makes a concrete movement towards solving the important security problem facing many P2P users. Also, our solution is scalable and easy to deploy by leveraging the existing P2P infrastructure, without involving a dedicated Content Distribution Network (CDN) (e.g., Akamai).

Organization: The rest of the paper is organized as follows. Section 2 describes an attack model for the network, as well as the design principles and the assumptions. Section 3 and 4 introduce two internal patching schemes for securing participating hosts in P2P systems. We evaluate the schemes in Section 6 and describe the related work in Section 7. We conclude in Section 8.

2 Preliminaries

Network Model. Many P2P file-sharing systems are actively running in these days (a comparison can be found in [2]). The most popular ones include eMule, KaZza, Gnutella, and BitTorrent. For concreteness, however, our discussion will focus on those unstructured networks such as Gnutella and KaZza. In Section 5 we will briefly mention how our approaches can be extended to other P2P systems such as BitTorrent.

We use Gnutella as an example to describe the file-sharing process. Specifically, each node uses a *shared folder* to store those files it wishes to share. When a requesting node initiates a download request for a specific file, it places a search for the target node(s) responsible for the given file identifier. The search request is routed through a two-tiered system of ultra-peers and leave nodes in the Gnutella overlay. In response, the requester collects a list of peers, each of which contains a file copy (probably with different versions). The requester then connects to one target node in the list and downloads the copy. Finally, she opens the downloaded file for use.

Attack Model. A file-sharing worm usually copies itself to a host's shared folder and publishes it with an attractive name, for example, as a popular song or movie. Sometimes attackers replace real movie or sound files with their malicious copies or add executable extensions to such files. When a host searches for some file and finds a match from an infected machine, it downloads and opens the file without being aware of the threat. Consequently, the worm is activated and it copies/attaches itself to all the files in the shared folder(s) of this new victim. In this way, a file-sharing worm continues its spread cycle.

We define two states for a file in P2P systems: *normal* and *abnormal*. A file is normal when it is valid and clean, and it becomes abnormal once malicious codes have been injected or attached to it. Also, we define three states with respect to a surging worm for each host in the systems: *vulnerable*, *infected*, and *immune*. A vulnerable host is not well-protected against the worm, hence it gets infected when exposed to the attack. For example, when a user opens a downloaded file which is abnormal, all files inside the shared folder(s) of this infected machine consequently become abnormal. A vulnerable/infected node becomes immunized once the protection (e.g., a patch) has been in place. Fig. 1 illustrates the node state transitions. We note that in real applications, some P2P users could voluntarily install the vendor patch on their machines. Therefore, these nodes are initially immunized to the worm. For simplicity, we assume no such individual recoveries occur during the period of defense.

We notice that a few elaborated worms such as Worm.Win32.Hofox were recently reported to be able to block the anti-virus protection services or kill anti-virus programs on P2P hosts. Clearly, at the system level, some local countermeasures will be devised to protect defense tools from being eliminated, and the arms race will continue. In this paper, however, we assume that P2P worms cannot disable the patching protocol

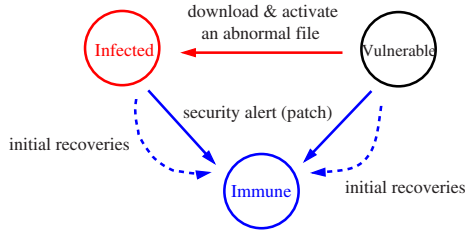


Fig. 1. Node state transition during the defense of internal patching. Initial recoveries include individual updates from security vendors; its percentage is relatively low as a new worm surges.

deployed in end hosts, so that an infected host can receive patches and become immunized as it is expected. Note that this assumption will not affect the correctness of our approaches, and our analysis model presented later can be slightly changed for the case when this assumption does not hold.

System Design Overview. To effectively combat file-sharing worms, we cannot merely rely on users' precaution and worm elimination skills; rather, we need *an automated and systematic approach to disseminate security patches to the P2P users*. Existing automated patching systems can be utilized to secure P2P hosts as well by simply treating them as normal Internet hosts; however, they are not necessarily the best-fit choices because not all Internet hosts are equally exposed to those P2P worms. For P2P users who often download files, their machines are more likely to be affected by P2P worms, whereas for non-P2P users, their machines will not be affected by worms exploiting P2P applications. Moreover, a traditional centralized model of patch distribution could cause single-point failure or overloading on the patch servers.

As such, we are motivated to study a P2P-tailored automated patching mechanism as a supplement to existing solutions and examine its effectiveness. Our approach utilizes the existing file-sharing infrastructure to internally push security updates (alerts) to the participating peers. It has several good features. First, it is customized for P2P environments and delivers security updates to P2P hosts only. This avoids unnecessary consumption of network/computer resources. Second, it adopts a distributed manner to disseminate security patches to those vulnerable peers in need and no longer strains the central servers. Third, our push-based scheme delivers security updates more promptly than the traditional once-a-day update adopted by existing patching systems.

Internal patching should leverage the existing file-sharing infrastructure for distributing security patches. Approaches utilizing IP address scanning or topology exploration to locate alive patching targets bring extra computation and communication overhead to P2P systems. Moreover, these methods could be easily exploited by malicious users and used as the vehicle for rapid worm spread and denial-of-service attacks.

In this paper, we study two push-based patching mechanisms for P2P systems. We first examine a download-based approach, in which a small fraction of popular nodes (also referred as *key nodes*) act as early patch distributors and a node which downloads a file from a key node will also be offered with a security patch/alert. Thus, the patch is propagated to many active hosts along with the file-download process. We then

examine a search-based approach, in which once a key node detects worm infection in a downloaded file, it re-performs a file search to identify those active hosts that possibly possess the abnormal file and disseminates the patch to immunize/disinfect them.

3 A Download-Based Approach

In our download-based approach, a small set of key nodes internally push the security patches to participating peers through the file-downloading process. Nodes that are notified of the approaching threat hence have a good chance of being immunized/disinfected against the worm. Our design of the scheme answers the following questions.

- Which hosts are determined as the key nodes so that they can distribute patches to others in a most efficient and timely manner? Key nodes cannot be determined in a centralized mode because no node in the system holds a global knowledge of file-sharing activities of others.
- What is a user’s strategy to choose the download source and what is the impact on patch dissemination? How should the existing P2P file transfer protocol be adapted to support the patch dissemination?
- How does a recipient authenticate a patch that it receives? In a distributed environment, even if a public key infrastructure (PKI) is deployed to provide sender authentication, it cannot prevent malicious peers from injecting worm codes instead of security patches. In other words, a node cannot fully trust others in the P2P system. Moreover, how does the recipient deal with the patch and how does her decision affect the immunity level of the system?

3.1 Scheme Description

Bootstrapping Key Nodes. The first important issue is the choice of key nodes. In most decentralized systems such as Gnutella and KaZaA, downloading traffic is highly focused around a small minority of popular targets and these popular files tend to be gradually concentrated in a small set of providers. For example, in Gnutella, 50% of all files are served by just 1% of nodes and 98% of all files are shared by the top 20% nodes [14]; in KaZaA, 10% most popular files generate 60% of the download traffic and 70% of the highly popular files will remain popular for at least 10~15 days [18]. These are strong indications that a small fraction of popular hosts sharing the most interesting files could be conveniently leveraged as the early distributors, which effectively push security patches to active downloaders in the system.

We consider a distributed algorithm for bootstrapping such key nodes in the P2P systems. Specifically, a small set of key nodes are individually decided according to a predefined policy. These key nodes then automatically pull (download and launch) the patch from vendor, so that they become immunized against the surging worm. To describe this algorithm in detail, we first introduce a node parameter named *file-offering rate* ϕ_O , which is defined as the number of files a node offers to its requesters in unit time. Note that this parameter reflects the popularity degree of the node. Each node calculates its ϕ_O based on its own file-sharing history. For example, node i may derive $\phi_O(i) = D_{out}(i)/T_f$, where $D_{out}(i)$ denotes i ’s out-degree in its file-access graph

within its neighborhood time window T_f . Thus, we can adopt the following policy to bootstrap the key nodes: *key nodes are selected from a subset of popular nodes with the highest file-offering rates in the system*. Specifically, each candidate node i refers to its recent file-offering rate $\phi_O(i)$ and decides to be a key node only if $\phi_O(i) \geq H_O$ satisfies. Here H_O is a globally defined threshold, which controls the fraction of key nodes. This policy can be automatically enforced through the client program. Once a node decides to become a key node, it should automatically fetch the latest security updates (if there are any) from the trusted vendor(s) and immediately launch the protection on the local machine (another option is they register to the vendors so that the vendors may push the latest security patch to them once available). Thus, key nodes get immunized against the surging worm and are ready to secure other file requesters. We note that as an active holder of more popular files, the user has to sacrifice a little bit convenience (patch activation if needed) and bandwidth (patch transfer) for the sake of the security of the entire system. On the other hand, a key node could be malicious or a regular node may claim to be a key node. We will discuss the related security issue shortly.

Disseminating Security Patches. Next, we discuss the message format of a security patch generated by the key node. This patch is used to notify the receivers of the worm threat and to provide the source of the security update. As illustrated in Fig 2, a patch message MSG_a typically contains two parts: a message header which contains the key nodes's identifier, and a message payload which contains (1) the worm alert (name, type, severity level, etc.), (2) the security patch itself (e.g., a Microsoft XP patch in binary delta compression format [6]) or simply a link to the URL of that patch (e.g., the Microsoft Security Bulletin), (3) a vendor signature of (1) and (2). We note that for a specific worm (identified by a specific vendor), the payload of its security patch message is unique. In addition, the security patch is *self-verifiable*: either the signature from a well-known vendor is attached to the patch, or the patch link can be directly verified through the vendor's website. This mechanism does not require a recipient to authenticate the intermediate patch distributors. Instead, it verifies the authenticity of the message content with the vendor or through its web site – these are considered more reliable and trustable.

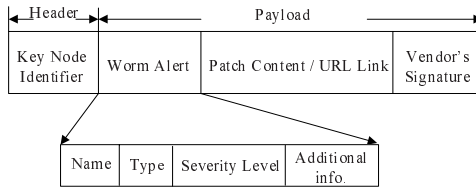


Fig. 2. Message format of security patch MSG_a

The next issue is how key nodes leverage the existing P2P file transfer protocol to internally distribute the security patch to file downloaders. Using Gnutella 0.6 system as an example, search results are directly delivered to a client (requester) through UDP packets. If the client chooses a resulting node for file download, it typically sends an HTTP Request to the provider and reads the bit stream of file content that follows

the HTTP Response [1]. We propose that the client also includes its latest patch version in the HTTP Request. Thus, a key node verifies the request before its file transfer and decide whether to distribute the latest security patch. To notify the recipient of the patch existence, the key node simply sets an indicator and the patch length in its HTTP Response. Both sides may establish an additional channel or use the existing control channel to perform patch transfer. In the case when the provider is behind a firewall, a PUSH process is executed to establish the connection [1] and the remainder of the file download and patch dissemination is similar as we have described.

Client Strategy and User Behavior. In response to a file search, a client receives a set of replies pointing to different file providers. The main decision that the client needs to make is which one of these node to ask for a copy of the file. This choice clearly has some influence on the defense. We consider the following three major selection strategies:

Random. The client selects a random node, independent of the node's advertised resources. In this mode, when there are α percentage of key nodes in the system, every client has an equal chance of α to download the file from a key node. This also implies that every client will eventually receive the security patch from key nodes.

Best. The client selects the node that advertises the best performance, i.e., the node with the lowest estimated delay (the node's queue length times the file size times the maximum number of simultaneous uploads divided by the access link bandwidth). Unlike the random mode, in this mode every client has a higher chance of downloading files from key nodes (i.e., those popular file holders). However, a small fraction of nodes which are not interested in the popular files may not have a chance to receive the security patch from key nodes.

Redundant. The client performs redundant download from either randomly chosen C nodes or C nodes with the lowest estimated delay. Once the first download finished and the content is verified for correctness, the other downloads are stopped. When the file download from a key node is aborted, the client cannot receive the patch that follows.

Next, we discuss how recipients react to the security patch. Although our scheme effectively leverages the internal infrastructure to expedite patch dissemination and ensures most participating peers receive the update as the file downloads proceed, the immunity level of the system is still in some degree dependent on individual users' responses to the patch. Upon receiving a patch message MSG_a , the client program first examines the payload and compares it with the existing version in order to discard out-dated or duplicated patch content. An accepted patch notifies the user of a surging worm and reminds her to launch immediate protection. If the user accepts the patch, the client program authenticates the patch payload either by directly examining the vendor signature or by visiting the trusted vendor site and verifying if the patch link provides consistent worm information. The program applies the new patch (a download is possibly needed) on the local machine immediately after a successful verification. Thus, the local machine gets immunized/disinfected against the worm and consequently all the files in its shared folder are/become normal. However, when a user declines the offer, either unwilling to follow the link or failing to activate the patch, her machine remains

Table 1. Notation for worm propagation model

Note.	Explanation
N	the total number of hosts in the network
$V(t)$	the number of vulnerable hosts
$I_f(t)$	the number of infected hosts
$I_m(t)$	the number of immune hosts
$F(t)$	the total number of files
$h(t)$	the proportion of abnormal files in the system
$s(t)$	the average size of a shared folder
λ_d	the average rate of file download (files/hour)
λ_a	the probability a user activates the downloaded file
α	the percentage of key nodes in the system
β	the probability at which a user accepts a patch

vulnerable to the worm. We quantitatively analyze the impact of user behavior on the system immunity in Section 3.2

3.2 Security and Performance Analysis

We derive a new fluid model for worm propagation and analyze the security and performance of the download-based approach. We refer to notation in Table 1.

A Fluid Model for Worm Propagation. We first consider the case when no defense has been deployed in the system. Each node is either in vulnerable or immune state, i.e., relation $N = V(t) + I_f(t)$ always satisfies. We show the evolution status of the system under the worm threat. The vulnerable population decreases as some nodes unfortunately download abnormal files, activate these files and get infected. We have

$$\frac{dV(t)}{dt} = -\lambda_d \lambda_a \cdot V(t) \cdot h(t). \quad (1)$$

Here $1/\lambda_d$ is the average time a node takes to download a file, and $h(t)$ reflects the percentage of abnormal files at time t . Solving the above differential equation, we get

$$V(t) = N - I_f(t) = V(0) \cdot e^{-\lambda_d \lambda_a \int_0^t h(\tau) d\tau}, \quad (2)$$

where $V(0)$ denotes the initial number of vulnerable hosts. This equation indicates that the vulnerable population in the system decreases exponentially as there are more file downloads and activations; the increase of the proportion of abnormal files accelerates the worm spread. We further derive the file state.

Lemma 1. *In a P2P file-sharing system, the percentage of abnormal files can be computed as $h(t) = h(0) \cdot e^{\frac{\lambda_d \lambda_a}{N} \int_0^t V(\tau) d\tau}$, where $h(0)$ is the initial abnormal rate. An approximation of $h(t)$ can be computed as $h(t) \approx \frac{I_f(t)}{N}$, assuming $\lambda_a \rightarrow 1$.*

This lemma is proved in Appendix A. It shows that user behavior has significant impact on the percentage of abnormal files: more file downloads and activations lead to more infections. However, as the amount of files increases and the vulnerable population decreases, worm infection is gradually slowed down.

Analysis of the Download-Based Defense. Time Performance Next, we examine the download-based defense. For simplicity, we assume users always adopt the random strategy to choose file providers (see 3.1) and all infected hosts have a patch acceptance probability β . We define *immunity rate* $i(t)$ as the fraction of immune nodes $i(t) = I_m(t)/N$, and let the initial immune population be $I_m(0)$. Note that these nodes, including the key nodes, either have applied the patch or does not expose the software vulnerability to the worm.

To study how long it takes to achieve a certain level of immunity rate, we formalize the problem as finding a lower bound t_0 for time t , so that we have $i(t) \geq \Psi$ when $t \geq t_0$, where $\frac{I_m(0)}{N} \leq \Psi \leq 1$ is a predefined threshold.

Lemma 2. *In a file-sharing system which adopts the download-based defense, the number of immune nodes is $I_m(t) = N + (I_m(0) - N) \cdot e^{-\lambda_d \alpha \beta t}$ and the system takes at least $t_0 = \frac{1}{\alpha \beta \lambda_d} \ln \frac{N - I_m(0)}{N(1 - \Psi)}$ hours to achieve an immunity rate Ψ .*

Proof. From the state diagram in Fig 1, we know that $N = V(t) + I_f(t) + I_m(t)$ always holds. Each time when a node downloads a file from a key node, it also receives a patch and the user decides whether or not to accept it. Note that only infected and vulnerable ($I_f(t) + V(t)$) nodes are immunized/disinfected in this process. We derive the change of immunity rate.

$$\frac{dI_m(t)}{dt} = (I_f(t) + V(t))\lambda_d\alpha\beta = (N - I_m(t)) \cdot \lambda_d\alpha\beta. \quad (3)$$

Here α also denotes the probability that each client selects a key node as the provider. Solving this differential equation for $I_m(t)$, we get the number of immune nodes in the system

$$I_m(t) = N + (I_m(0) - N) \cdot e^{-\lambda_d \alpha \beta t}. \quad (4)$$

From the given condition $i(t) = I_m(t)/N \geq \Psi$, we may further derive $t \geq t_0 = \frac{1}{\alpha \beta \lambda_d} \ln \frac{N - I_m(0)}{N(1 - \Psi)}$.

Fig 3 illustrates the change of immunity rate $i(t)$ when the percentage of key nodes (α) varies from 5% to 15%. Clearly, as there are more patch distributors, the system takes less time to reach a certain level of immunity rate (in our case 90%). For example, when $\alpha = 5\%$, it takes 60 hours for 90% of nodes to receive the patch, whereas it takes 20 hours when $\alpha = 15\%$. This figure also shows that in the random selection mode, each downloader (including those not interested in the popular files) will eventually receive the patch from a key node.

System Evolution Status. We also examine the evolution status of the system which adopts the download-based defense. During the worm containment, a vulnerable host either (1) becomes infected when it downloads and activates an abnormal file from a

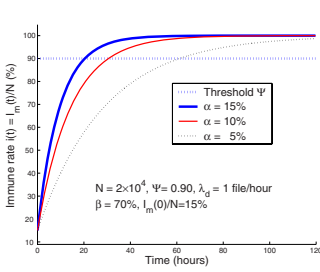


Fig. 3. Immunity rate as a function of time and key nodes

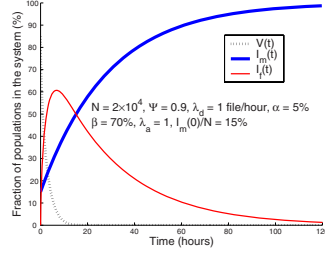


Fig. 4. System evolution status under the download-based defense

non-key node, or (2) gets immunized when it downloads a file from a key node and accepts the patch. Hence, we set up the following equation for the change of the vulnerable population.

$$\frac{dV(t)}{dt} = -\lambda_d \lambda_a (1 - \alpha) \cdot V(t) \cdot h(t) - \lambda_d \alpha \beta \cdot V(t). \quad (5)$$

Note that here $\lambda_d \lambda_a (1 - \alpha) \cdot V(t) \cdot h(t)$ computes the reduction caused by (1) ($1 - \alpha$ denotes the probability of downloading from a non-key nodes) and $\lambda_d \alpha \beta \cdot V(t)$ computes the reduction caused by (2). We use the approximation $h(t) \approx \frac{I_f(t)}{N}$ and the solution in Equ 4 to solve this differential equation for $V(t)$. We also compute $I_f(t)$ using the relation $N - I_m(t) - V(t)$. Fig 4 illustrates the evolution status of a file-sharing system. Initially, the percentage of infected nodes increases as the worm surges. However, when more and more file downloaders receive the patch, worm infections are gradually cleansed from the network and the infected population starts to decrease. Eventually, immune nodes become the major population. The figures also indicate that the immune time t_0 is determined by several factors: the fraction of the key nodes (α), the file downloading rate (λ_d), patch acceptance rate (β) and the initial immunity rate. Our analytical result has been validated in Section 6 (Fig 8).

4 A Search-Based Approach

This section proposes a search-based approach, in which once a key node detects worm infection in a file it has just downloaded from other participating peers, it immediately infer from a new search result a set of suspicious targets, to which it directly pushes the security patch and disinfect/immunize them. Given the latest vendor updates, we assume key nodes are able to detect on-going worm attacks based on techniques such as worm signature matching, taint analysis or anomaly detection [9][16][20][19]. We answer the following questions in our design.

- Which hosts in the system should be chosen as the key nodes, so that they detect file anomalies in the system and distribute patches to others in a most efficient and timely manner? Key nodes should be bootstrapped in a distributed way.

- Once a key node detects file anomaly, how does it infer a set of suspicious nodes by examining the query response and how to deal with network dynamics?
- How does a key node disseminate the patch to those suspicious nodes? To be scalable, how should a key node limit its bandwidth for patch delivery?
- What is the user’s reaction towards the patch and how does it influence the immunity level of the system?

4.1 Scheme Description

Bootstrapping Key Nodes. To address the first issue, we consider a distributed algorithm to bootstrap key nodes in our search-based scheme. Similar to the algorithm in Section 3.1, key nodes automatically pull (download and install) the latest vendor patch so that they become immunized against the surging worm. However, here we adopt a different policy for determining key nodes in a distributed way. We first introduce a node parameter named *file-downloading rate* ϕ_I , which is defined as the number of files a node downloads from others in unit time. This parameter reflects a node’s activity level of file downloads. Each node i derives $\phi_I(i) = D_{in}(i)/T_f$, where $D_{in}(i)$ denotes the number of files i has downloaded within the time window T_f . Now we can adopt the following policy to bootstrap the key nodes: *key nodes are selected among a subset of nodes with the highest file-downloading rates in the system*. Specifically, each candidate node i refers to its recent file-downloading rate $\phi_I(i)$ and decides to be a key node only if $\phi_I(i) \geq H_D$ satisfies. Here H_D is a globally defined threshold which controls the fraction of key nodes. This bootstrapping policy is automatically enforced through the client program within an end host.

The above policy chooses those active file requesters as the key nodes because these nodes keep actively acquiring files from various origins, hence their chance of being infected is relatively higher than hosts with relatively low downloading rate. Keeping these nodes updated with the latest vendor patch also enables them to explore more worm infections from file providers. Our search-based scheme requires a key node P immediately examines the file state after it finishes downloading a file f_p . Once an anomaly has been identified, the key node composes a security patch message Msg_a .

Distributing Security Patches. The next issue is to which nodes the security patch should be distributed. Pushing the patch directly to the provider who has uploaded the abnormal file is effective. However, this is not efficient because the key node has a good reason to suspect that other file-owners may have also been infected. On the other hand, simply flooding the patch or locating the targets by IP address scanning or topology exploration is not scalable. Our solution is to let the key nodes exploit the file search list to locate those *suspicious* file providers and push the patch to these targets. However, there exist a time gap (could be in hours) between the original search and the worm detection. During this period, nodes frequently join and leave the network. A good strategy for the key node is to re-perform a file search once it has detected a file anomaly.

We propose a distributed algorithm for patch dissemination, as illustrated in Fig 5. Specifically, once a key node P has detected worm infection in a downloaded file f_p , it immediately re-perform a search on f_p and consequently receives multiple QueryHit responses, based on which it sorts the destination nodes according to the *activity level* and constructs a *ranked* search list S_p . Here a node i ’s activity level $L_a(i)$ is derived from

three parameters: the bandwidth of its access link $Spd(i)$, the queue length $QLen(i)$ and the number of simultaneous uploads $N_{up}(i)$. These parameters for node i are included in the QueryHit message, i.e., we have $L_a(i) = f(Spd(i), QLen(i), N_{up}(i))$, where f is a monotonically increasing function. The application in the key node then computes an activity lower bound $H_L(P)$ based on the bandwidth $Spd(P)$ and the current number of connections (ongoing P2P traffic) in the local machine. Finally, the key node chooses from S_p top k target nodes whose activity level satisfies $L_a \geq H_L(P)$ and establishes a direct HTTP connection with each of these suspicious targets to push the security patch. Note that such patch transfers are out-of-band (not through the Gnutella overlay).

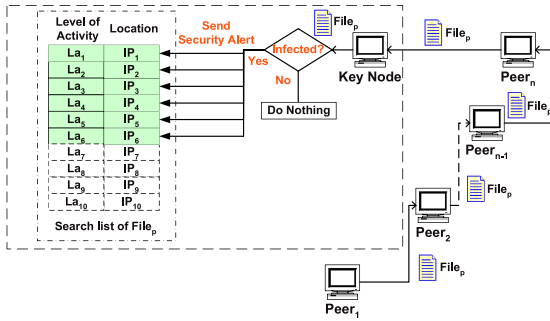


Fig. 5. An illustration of the search-based approach. In this example, key node P detects an infected file f_p and delivers patch MSG'_a to $k = 6$ suspicious nodes in the search list S_p .

Here a security patch MSG'_a also contains the identifier of the infected file f_p . Upon receiving the patch, each node j needs to verify the existence of f_p and then displays a warning in the local machine. If the user accepts the patch, the application first authenticates the patch content/link. Once the patch has been successfully applied, node j becomes immunized to the worm and its shared folder will be immediately scanned and cleansed. We quantitatively analyze the impact of user behavior on the system immunity level in Section 4.2.

4.2 Security and Performance Analysis

Time Performance. We analyze the effectiveness of the search-based defense. Let k denote the average number of suspicious targets to which a key node distributes the security patch, we first derive how long the system takes to achieve an immunity rate Ψ . According to the state diagram in Fig 1, $N = V(t) + I_f(t) + I_m(t)$ always holds. In the search-based scheme, the increased immune population comes from either vulnerable nodes or infected nodes. Hence, the rate at which the immune population increases can be computed as

$$\begin{aligned} \frac{dI_m(t)}{dt} &= N\lambda_d \cdot \alpha h(t) \left(\frac{N - I_m(t)}{N} \right) \cdot k\beta \\ &= a(N - I_m(t)) \cdot h(t), \end{aligned} \tag{6}$$

Where $a = \alpha\beta\lambda_d k$. Note that $\alpha h(t)(\frac{N-I_m(t)}{N})$ computes the probability that a key node downloads an abnormal file from those non-immune hosts.

Also, the decrease of the vulnerable population could be caused either by (1) worm infections or (2) by host immunizations (or disinfections). Therefore, the rate at which vulnerable nodes become either infected or immunized is

$$\begin{aligned} \frac{dV(t)}{dt} &= -\lambda_a \lambda_d V(t)h(t) - N\lambda_d \cdot \alpha h(t)(\frac{V(t)}{N}) \cdot k\beta \\ &= -(a+b)V(t)h(t). \end{aligned} \quad (7)$$

where $b = \lambda_a \lambda_d$. Here $\lambda_a \lambda_d V(t)h(t)$ computes the reduction caused by (1); $N\lambda_d \cdot \alpha h(t)(\frac{V(t)}{N}) \cdot k\beta$ computes the reduction caused by (2), in which $\alpha h(t)(\frac{V(t)}{N})$ denotes the probability a key node downloads an abnormal file from a vulnerable host (not infected yet). In this case, the latter receives the patch and could be immunized.

Finally, we know that the infected population (1) increases when some vulnerable nodes get infected, and (2) decreases when some victim nodes have been disinfected. Hence we derive the following differential equation.

$$\begin{aligned} \frac{dI_f(t)}{dt} &= \lambda_a \lambda_d V(t)h(t) - N\lambda_d \cdot \alpha h(t)(\frac{I_f(t)}{N}) \cdot k\beta \\ &= bV(t)h(t) - aI_f(t)h(t). \end{aligned} \quad (8)$$

Note that here $\lambda_a \lambda_d V(t)h(t)$ computes the increase of infected nodes caused by (1); $N\lambda_d \cdot \alpha h(t)(\frac{I_f(t)}{N}) \cdot k\beta$ computes the reduction of infected nodes caused by (2), where $\alpha h(t)(\frac{I_f(t)}{N})$ denotes the probability a key node downloads an abnormal file from an infected host. In this case, the latter receives the patch and could be disinfected. To solve these differential equations, we derive the immune population $I_m(t)$. We divide Equ. 6 by Equ. 7 and get

$$V(t) = V_0^{-\frac{b}{a}} \cdot (N - I_m(t))^{\frac{a+b}{a}}, \quad (9)$$

where $V_0 = V(0)$ denotes the initial vulnerable population in the system. We then apply the approximation $h(t) \approx \frac{I_f(t)}{N} = \frac{N-I_m(t)-V(t)}{N}$ and substitute Equ. 9 into Equ. 6. Thus, we have

$$\frac{du}{dt} = -\frac{a \cdot V_0}{N} \cdot u^2(1 - u^{\frac{b}{a}}) \quad (10)$$

where $u = (N - I_m(t))/V_0$. We further solve this equation for $I_m(t)$ and illustrate the change of $I_m(t)$ in Fig. 6. This figure indicates that under an average size $k = 30$, the search-based approach only needs to deploy 5% nodes as key nodes and help the system achieve a 90% immunity rate within 60 hours.

System Evolution Status. Adopting the similar method as above to solve Equ. 6, 7, 8, we derive the following

$$\frac{dV(t)}{dt} = -\frac{a+b}{N}(c \cdot V^{1+\frac{a}{a+b}}(t) - V^2(t)), \quad (11)$$

where $c = V_0^{\frac{b}{a+b}}$. Hence we may compute $V(t)$. Using $I_f(t) = N - V(t) - I_m(t)$, we may further derive the infected population $I_f(t)$. We illustrate the system evolution

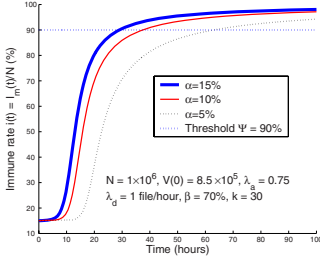


Fig. 6. Immune population vs. time and key nodes

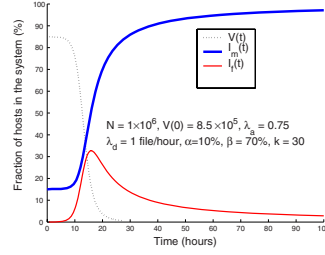


Fig. 7. System evolution status in search-based scheme

status in Fig 7. The figure shows that the infected population initially increases due to the surging worm. However, this triggers the defense and many suspicious nodes receive the patch. Eventually worm infections are eliminated and immune nodes become the major population. The above analytical result has been validated in Section 6 (Fig 8).

5 Security Analysis

We discuss two attacks that may happen in both schemes.

Fake Security Alerts. A malicious node, either a key node or a regular node claiming to be a key node, may replace security patches with worms and deliver them to other hosts. This attack will fail because our signature-based mechanism allows a receiver to verify if the patch truly comes from a trusted vendor or the link to the patch is correct. On the other hand, we notice that a lot of false messages may cause a DoS attack to other hosts. Since we do not assume a PKI, P2P nodes may not be able to authenticate each other. Indeed, even a PKI is available, it does not solve this type of insider attacks. A simple solution is that a node blacklists the nodes reporting false alerts based on their IP addresses. To prevent IP spoofing, before a node accepts a security alert, it challenges the source.

Patch Suppression Attack. A malicious (or selfish) candidate key node may not propagate security patches. That is, in the download-based approach, it does not offer the security patches to downloaders and in the search-based scheme it does not care about other susceptible nodes. This patch suppression attack will degrade the effectiveness of our schemes. However, it only decreases the actual α . As long as they are not a lot, our schemes will still work. Otherwise, we should increase the value of α .

6 Evaluation

Environmental Setup. We evaluated and compared our schemes in a variety of file-sharing systems. For unstructured networks we implemented a Gnutella simulator based on Gnutellasim from limewire.org; for structured networks we used P2PSim ([5]) to

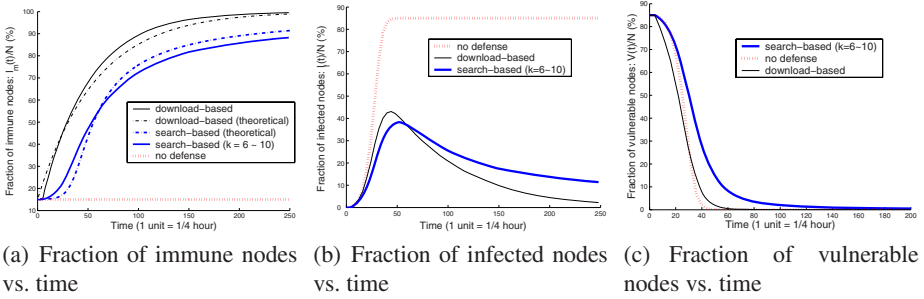


Fig. 8. Comparing time performance of the approaches in Gnutella 0.6; $N=20k$ nodes, $M=1k$ files, $\lambda_d=1$ file/hour, $\lambda_a=1.0$, $\alpha=10\%$, $\beta=0.7$

construct a basic Chord [22] infrastructure for routing queries and responses. We implemented a protocol similar as NeuroGrid [15] to generate large-scale file-sharing traffic on top of the routing infrastructures. We studied the case when a file-sharing worm Benjamin.a [4] surges in the network and evaluated the effectiveness of our countermeasures.

We adopted the following metrics in our evaluations: t_0 , time takes to reach an immunity rate $\Psi = 90\%$; $h(t_0)$, the the percentage of abnormal files at time t_0 , which also reflects the infection rate $I_f/N(t_0)$ when $\lambda_a \rightarrow 1$; $I_f(max)$, the maximum infection rate which indicates how severe the system has been attacked. For each scheme, we also investigated the system evolution status, the impacts from user behavior and the message overhead. To examine the schemes' tolerance against node dynamics (joins/departures), our implementation followed the observations from Gnutella 0.6, i.e., 45% of the nodes quit the network in less than 4 ~ 5 hours, and 22% persistent node tend to stay in the network for longer than 24 hours. Each of our experiments takes 100 runs. We report the mean of the measurement results. Unless otherwise indicated, in all our tests, the total population $N = 20,000$ nodes. The number of files (with different contents) varies from 1,000 to 10,000 and the average size of shared folders ranges from 5 to 50 files. We set the initial the percentage of abnormal files $h(0) = 1.5\%$, the initial infection $I_f(0)/N = 0$ and the initial immunity rate $i(0) = 15\%$. Among these immune nodes, $\alpha = 5 \sim 10\%$ of the entire population were bootstrapped as key nodes and each of them obtained the latest security updates from vendors.

Scheme Effectiveness. We compared the time performance and the system evolution status of different approaches, using the same set of parameters (e.g., λ_a , λ_d , α and β). We also used the no-defense case as the base line. Our test results are shown in Fig 8. Fig 8(a) illustrates the change of immune population over time. Without any defenses, the system keeps a low immunity rate and has to rely on individuals' patch updates. The download-based approach and the search-based approach both significantly increase the immunized population. The former takes around 35.5 hours to achieve a 90% immunity rate while the latter takes around 62.5 hour due to its reactive nature. The download-based approach largely depends on the activity level of file downloads and the search-based approach is triggered by worm detections. Fig 8(b) shows the change of the infected population over time. Without any defenses, the worm spreads

in a relatively high speed and infects all the vulnerable hosts within 9.5 hours. Both our schemes effectively help the system reduce the infected population by internally pushing the security patch to disinfect those victims. A further comparison indicates that the search-based approach has a relatively slower disinfection speed; it takes 62.5 hours to reduce the overall infection rate to below 10%. However, it keeps a lower maximum infection rate ($I_f(max)/N = 37%$). On the contrary, the download-based approach takes 45 hours to reach an infection rate below 5%, but it yields a higher maximum infection rate ($I_f(max)/N = 44%$) in the system. Fig 8(c) illustrates the change of vulnerable population over time. Without any defenses, the vulnerable population quickly drops to zero (within 10 hours) as more and more nodes get infected during file downloads. Our schemes effectively slow down this process by either immunizing the vulnerable hosts or disinfecting the victims. We can see that after around 62.5 hours, there remain few vulnerable nodes and victims in the system and the immunity rate exceeds 90%.

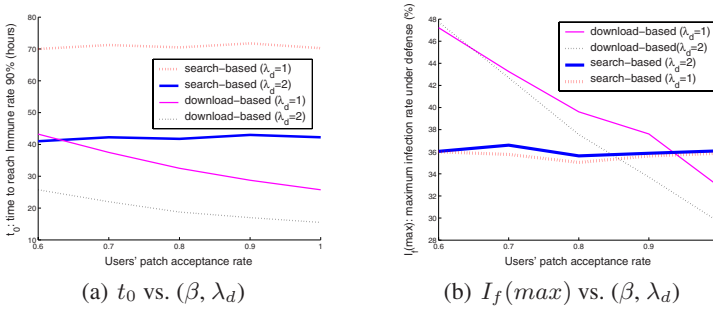


Fig. 9. Impact from user behavior on the defense scheme (Gnutella 0.6)

User Impacts and Overhead. We evaluated the impacts of system parameters and user behavior on the defense. Fig 9 illustrates the test result in Gnutella 0.6 system. Fig 9(a) shows that both schemes take less time (t_0) to achieve a 90% immunity rate as the speed of file download (λ_d) increases. For the download-based approach, a higher download speed results in a faster patching process; for the search-based approach, a higher download speed leads to more worm infections and this in turn speeds up the patching process. The figure also indicates that in the download-based approach, t_0 gets reduced as users become more willing to accept the patch (β increases). However, this is not distinct in the search-based scheme due to its reactive nature. When more users are patched, the defense also gets slowed down. Fig 9(b) shows that in the download-based scheme, the severity level of worm attacks ($I_f(max)$) quickly drops as β increases. When $\beta \geq 0.85$, the maximum infection rate in the system is below that of the search-based scheme.

Fig 10 illustrates the message overhead of the defense schemes. Using Microsoft XP as an example, the patches during SP2 are in binary delta compression format [6] and the mean patch size is 32.9 KBytes [12]. This patch and its vendor signature (typically around 300 Bytes) constitute the main part of the payload in an alert message. Thus, the average length of a patch message is 33.2 KBytes. The figure shows that when β

increases from 0.6 to 1.0, the message count of the download-based approach decreases until finally it reaches around $20,000 \times (90\% - 15\%) = 15,000$. We examined three cases for the search-based scheme: (1) *worst case* in which each key node simply delivers the patch to its targets. Patch messages could be duplicated and the message count is above 50,000; (2) *average case* in which a key node does not deliver a patch to the same target and the message count is above 28,000; (3) *optimal case* when key nodes collaborate to avoid patch duplicates or each node indicates its current patch version in the QueryHit response. Hence, there are few patch duplicates and the message count approaches 15,000 when $\beta \rightarrow 1$.

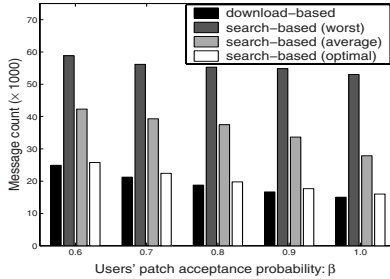


Fig. 10. Message overhead of the defense scheme ($N = 20,000$ nodes, $\Psi = 90\%$); the download-based approach has less overhead than the search-based one due to no alert duplicates.

7 Related Work

File-sharing worms have recently attracted much attention in research community. The initial studies mainly focused on understanding the threats and modeling the behavior of such imminent worms. Dimitriu et al. [11] demonstrated that worm spreads highly depend on user behavior, such as willingness to share files and quickness in removing infected files. Kumar et al. [17] developed a suite of fluid models that characterize pollution proliferation in file-sharing systems. Thommes et al. [23] derived deterministic epidemiological models for file-sharing viruses spreading in file-sharing systems.

The problem of throttling these worms have not been adequately addressed. Existing defenses mirror the strategy against Internet computer viruses. Generic automated patching tools (e.g., Microsoft Window Update, Symantec Update, McAfee VirusScan) are widely adopted to launch protection on P2P hosts. Vojnovic et al. [24] studied the effectiveness of automatic patching and quantified the speed of patch dissemination required for worm containment. Gkantsidis et al. [12] provided general guidelines on how to design a fast planet-scale patching system based on their studies on Window Update. They also suggested alternative patching strategies such as caching. Costa et al. [10] proposed Vigilante, an end-to-end approach in which hosts run instrumented software to detect worms and broadcast self-certifying alerts (SCA) upon worm detection. Zhou et al. [26] further applied Vigilante in P2P systems to contain fast-spreading topological worms. Our work differs from the above in that we provide internal patching mechanisms exclusively for file-sharing systems. Our focus is not on generating anti-worm

code on-the-fly to combat zero-day worms, but on studying good patching schemes which both save network bandwidth and avoid unnecessary host interruptions.

8 Conclusions and Future Work

File-sharing worms are becoming the most dominating and devastating security threats to P2P systems. Current defenses relying on individual recoveries or limiting file-sharing activities are not adequate. As a complement to the existing centralized patching mode, we proposed internal patching mechanism which conveniently leverages file-sharing infrastructure to disseminate security updates to participating peers in an automated and distributed way. We studied a download-based approach which exploits the file download process and a search-based approach which exploits the file search process for notifying P2P hosts of the worm attack and pushing the security patch to them. In spite of some remaining issues such as host diversity and user diversities, the free-rider problem [8] in patching, our study suggests some interesting directions for designing countermeasures against worms in distributed environments. We address remaining issues in our future work.

Acknowledgements

This research was supported in part by the National Science Foundation CAREER-0643906. We thank the anonymous reviewers for their helpful comments.

References

1. The gnutella protocol specification, <http://www.the-gdf.org>
2. http://en.wikipedia.org/wiki/comparison_of_file_sharing_applications
3. <http://www.facetime.com/securitylabs/imp2pthreats.aspx>
4. <http://www.viruslist.com/en/viruslist.html?id=49790>
5. P2PSim: a simulator for peer-to-peer protocols, <http://pdos.csail.mit.edu/p2psim>
6. Using binary delta compression technology to update windows operating systems. Microsoft online White Paper
7. www.viruslist.com/en/virusesdescribed?chapter=153311928
8. Biddle, P., England, P., Peinado, M., Willman, B.: The darknet and the future of content distribution. In: ACM Workshop on Digital Rights Management (2002)
9. Brumley, D., Newsome, J., Song, D., Wang, H., Jha, S.: Towards automatic generation of vulnerability-based signatures. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy (2006)
10. Costa, M., Crowcroft, J., Castro, M., Rowstron, A.: Can we contain internet worms? In: Proc. of the 3rd Workshop on Hot Topics in Networks (HotNets-III) (November 2004)
11. Dumitriu, D., Knightly, E., Kuzmanovic, A., Stoica, I., Zwaenepoel, W.: Denial-of-service resilience in peer-to-peer file sharing systems. In: Sigmetrics 2005 (2005)

12. Gkantsidis, C., Karagiannis, T., Rodriguez, P., Vojnovic, M.: Planet scale software updates. In: Proc. of SIGCOMM 2006 (2006)
13. Good, N., Krekelberg, A.: Usability and privacy: a study of kazaa p2p file-sharing (2002), <http://www.hpl.hp.com/shl/papers/kazaa/index.html>
14. Hughes, D., Coulson, G., Walkerdine, J.: Free riding on gnutella revisited: the bell tolls. In: Proc. of IEEE Distributed Systems Online (2005)
15. Joseph, S.: NeuroGrid: Semantically routing queries in peer-to-peer networks. In: International Workshop on Peer-to-Peer Computing (2002)
16. Kc, G., Keromytis, A., Prevelakis, V.: Countering code-injection attacks with instruction-set randomization. In: ACM CCS 2003 (October 2003)
17. Kumar, R., Yao, D., Bagchi, A., Ross, K., Rubenstein, D.: Fluid modeling of pollution proliferation in p2p networks. In: Sigmetrics 2006(2006)
18. Leibowitz, N., Ripeanu, M., Wierzbicki, A.: Deconstructing the kazaa network. In: Proc. of IEEE IWAPP 2003 (2003)
19. Mulz, D., Valeur, F., Kruegel, C., Vigna, G.: Anomalous system call detection. In: ACM TISSEC 2006 (2006)
20. Newstone, J., Song, D.: Dynamic taint analysis: Automatic detection and generation of software exploit attacks. In: Proc. of the 12th Annual Network and Distributed System Security Symposium (NDSS 2005) (2005)
21. Shin, S., Jung, J., Balakrishnan, H.: Malware prevalence in the kazaa file-sharing network. In: ACM Internet Measurement Conference 2006 (2006)
22. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M., Dabek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup protocol for internet applications. In: IEEE Trans. on Networking (2002)
23. Thommes, R., Coates, M.: Epidemiological modeling of peer-to-peer viruses and pollution. In: Infocom 2006 (2006)
24. Vojnovic, M., Ganesh, A.: On the race of worms, alerts and patches. In: ACM Workshop on WORM (2005)
25. Zetter, K.: Kazza delivers more than tunes. The Wired Magazine (2004)
26. Zhou, L., Zhang, L., McSherry, F., Immorlica, N., Costa, M., Chien, S.: A first look at p2p worms: threats and defenses. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, Springer, Heidelberg (2005)

A Percentage of Abnormal Files

Proof. We refer to Table 1 for notation. Let $A(t)$ denote the number of abnormal files at time t , we have $h(t) = A(t)/F(t)$. The change rate of the total number of files $F(t)$ is

$$\frac{dF(t)}{dt} = \lambda_d N \quad \text{i.e., } F(t) = F_0 + \lambda_d N \cdot t, \quad (12)$$

where F_0 is the initial number of files in the system. A newly added abnormal file could be caused either by a vulnerable host activating another abnormal file in the same folder, or by a node directly downloading an abnormal copy from others. Therefore, we have the change rate of the number of abnormal files

$$\frac{dA(t)}{dt} = \lambda_d \cdot \lambda_a \cdot V(t) \cdot h(t) \cdot (s(t) - 1) + N\lambda_d h(t). \quad (13)$$

Considering $s(t) - 1 \approx F(t)/N$ and $A(t) = F(t) \cdot h(t)$, we solve equation [13](#) and finally get

$$h(t) = h(0) \cdot e^{\frac{\lambda_d \cdot \lambda_a}{N} \int_0^t V(\tau) d\tau}, \quad (14)$$

where $h(0)$ is the initial percentage of abnormal files.

We may compute an approximation for $h(t)$. Assuming all the nodes have a similar size of $s(t)$ for their shared folders and the user parameter λ_a approaches 1, which means every client usually activates (opens) the file he has just downloaded, all the abnormal files should gradually be kept by those infected hosts in the system. Hence, we may derive $h(t) \approx \frac{I_f(t)}{N}$.

Peeking Through the Cloud: DNS-Based Estimation and Its Applications

Moheeb Abu Rajab¹, Fabian Monroe¹, Andreas Terzis¹, and Niels Provos²

¹ Johns Hopkins University, Baltimore MD 21218, USA
{moheeb,fabian,terzis}@cs.jhu.edu

² Google Inc., Mountain View, CA 94043, USA
niels@google.com

Abstract. Reliable network demographics are quickly becoming a much sought-after digital commodity. However, as the need for more refined Internet demographics has grown, so too has the tension between privacy and utility. Unfortunately, current techniques lean too much in favor of functional requirements over protecting the privacy of users. For example, the most prominent proposals for measuring the relative popularity of a website depend on the deployment of client-side measurement agents that are generally perceived as infringing on users' privacy, thereby limiting their wide scale adoption. Moreover, the client-side nature of these techniques also makes them susceptible to various manipulation tactics that undermine the integrity of their results. In this paper, we propose a new estimation technique that uses DNS cache probing to infer the density of clients accessing a given service. Compared to earlier techniques, our scheme is less invasive as it does not reveal user-specific traits, and is more robust against manipulation. We demonstrate the flexibility of our approach through two important security applications. First, we illustrate how our scheme can be used as a lightweight technique for measuring and verifying the relative popularity rank of different websites. Second, using data from several hundred botnets, we apply our technique to indirectly measure the infected population of this increasing Internet phenomenon.

Keywords: Client Density Estimation, Web-metering, Botnets, Network Security.

1 Introduction

Over the past few years, it has become increasingly important to garner reliable information about the demographics of the Internet and the myriad of services that it supports. For one, Internet businesses increasingly rely on such information to better customize their marketing campaigns. Advertisers, for example, make continual use of the relative popularity of websites and the demographics of their visitors to design and position their products and services in an effective manner. Similarly, security practitioners frequently use information about the

population affected by security incidents to develop a better understanding of the characteristics and the scope of these attacks.

However, as the need for network demographics has increased, so too has the tension between utility and privacy. For instance, the most well-known schemes for measuring the relative popularity of websites (e.g., Alexa [38], ComScore [20] and NetRatings [26]) collect client-side data from deployments of measurement agents placed inside edge networks (e.g., using browser toolbars that record all URLs visited by the client). Generally speaking, the collected data is sanitized and used to produce aggregate statistics for the application in question. However, since the type of sanitization that is applied, as well as the specific data collected, are completely at the discretion of the data collector, large cross-sections of the population shy away from deploying these agents. Moreover, the mere client-side nature of these collection schemes opens the door to abuse (be it via click fraud [23] or other manipulation recipes [1]) that directly affect the integrity of the results. Recently, the prevalence of these fraudulent behaviors has raised so much doubts about the integrity and authenticity of these ranking measures that it captured the attention of the mainstream press (e.g., [22]).

Also of much interest lately is the question of how to reliably determine the infected population of an all too common security event, namely, botnets. Clearly, the size of the population affected by a particular security incident plays an important role in fully understanding its impact, as well as helps in prioritizing defense tactics from industry and practitioners alike. Unfortunately, although information on the scale and nature of a security event can be valuable for forensic and defenses purposes, network operators are usually reluctant to release such information as disclosure of (repeated) breaches can lead to loss of public confidence. Therefore, information on the spread of security events (worms, botnets, etc.) is normally collected at a global scale by dedicated measurement entities (e.g., CAIDA) using a combination of direct [28,31] and indirect methods [3,24,31]. While these approaches have been widely successful, they are known to be vulnerable to various evasive tactics. For example, network monitors can easily be detected and evaded by active probing attacks [5,29,36]. Similarly, in the case of botnets, several practices complicate direct measurements as botmasters often suppress broadcast feedback, thereby making direct measurements infeasible even if the botnet has been infiltrated [30].

In what follows, we present a new technique for inferring the density of clients accessing a particular network service. Among a number of possible applications, our technique is directly applicable to both of the aforementioned problems. Specifically, we present an indirect estimation technique using DNS cache probing [14], and show how it can be used for website metering as well as for inferring the infected population of certain security events (e.g., botnets). In the former case, our evaluation shows that the technique is very accurate, and can serve as a standalone verification tool for determining the popularity rank of different websites. Compared to other approaches (e.g., Alexa [38]), our technique is less invasive as it does not require host-specific information. Moreover, as we discuss

later, our technique is more robust under a threat model where the attacker is deliberately trying to inflate the popularity rank of her website.

In the latter case, we illustrate how our technique can be used to arrive at a better size estimate (than the notion of botnet *footprint* we suggested earlier [31]). We argue that this refinement is important as botnets continue to be one of the top Internet threats today [32,37], and so more accurate size measurements have immediate benefit in assessing the monetary impact and damage they cause (e.g., via identity theft, DDoS attacks, etc.) [15]. While fine-grained estimates of botnet size still remain challenging [30], we believe this work offers a valuable step forward in that regard.

The remainder of the paper is organized as follows. In Sections 2 we illustrate our methodology and estimation techniques. In Section 3 we validate our approach through simulation and by comparison to an actual client count measured directly from our local network. In Section 4 we provide two real world applications that we believe aptly demonstrate the utility of our technique then we discuss some practical considerations in Section 5. In Section 6 we review related work, and conclude in Section 7.

2 Estimation Methodology

Growing security [18,34] and privacy concerns raise significant challenges for the application of direct methods to obtain faithful counts of clients using a particular service, for example, by simply taking measurements from within network boundaries (e.g., using toolbars that monitor a users' browsing habits). Rightfully so, this unease calls for indirect counting techniques that limit the privacy risks with recording host-specific information. In this section, we describe our methodology for estimating the number of clients accessing a particular network service using a purely indirect technique.

Our approach exploits the fact that most network services (e.g., websites, botnet command and control servers) use DNS names to identify their servers. This, in turn, makes DNS resolution a pre-requisite step for any client connecting to that server. Simply speaking, we exploit this association to infer the number of clients requesting the resolution of the DNS name for the service of interest (e.g., `www.cnn.com`) from their local DNS resolver. Specifically, we use DNS cache probing to measure the evolution of that name in the resolver's cache and consequently derive an estimate of the number of clients accessing that service. Compared to direct methods, our technique is less intrusive as it does not reveal the specific identities of clients accessing the service of interest.

The technique itself is rather straightforward: for each DNS name of interest, S , we probe the cache(s) of the DNS resolver(s) for the network(s) of interest at regular intervals and examine the observed cache hits, if any, for S . For each cache probe, a cooperative resolver (i.e., a resolver that responds to DNS cache queries) will report a hit if S was in its cache, or a miss otherwise. In the former case, the resolver also reports the remaining time before S is flushed. While a cache hit only indicates that *at least one* client made a request for that entry,

we can refine that estimate by sending appropriately-spaced probes that reveal the sequence of start and end times of S 's entry in the resolver's cache. These times are a direct result of the combined queries from all clients that the resolver serves.

Figure 1 illustrates our estimation methodology. At a high level, one can envision the client population estimation as involving two processes: an input process representing the combined arrivals of DNS queries for the DNS entry S , from clients served by the same resolver, and an output process representing the refresh and expiry times of S that result from the input querying process. Our subsequent analysis relies on the simplifying assumption that the inter-arrival times of client requests in the input process can be modeled as a sequence of independent identically distributed random variables (IID's). Jung *et al.* [21] also used this assumption and showed that it does not introduce significant bias in the arrival model. We will return to this assumption, as well as the arrival model, in Section 2.2. Given this model, our goal is to estimate the number of clients n , requesting lookups for S from their common DNS resolver. We do so by estimating the aggregate DNS query rate λ , using the observed refresh and expiry times of the entry S from the output process. As we show later, the resulting estimate for λ leads to an estimate of the number of clients n . In what follows, we begin by describing our methodology for estimating the rate λ and then proceed to show how we can use this estimate to infer the number of clients n in Section 2.2.

2.1 Estimating the Aggregate Rate

For a DNS entry S , with a time to live (TTL), we estimate the aggregate rate λ , as follows: we probe the cache of the resolver of interest at a rate of one probe per TTL . As Figure 1 illustrates, the sequence of probes allows us to capture the start and end times of S in the resolver's cache. Recall that for a cache hit corresponding to a probe p at time T_p , the resolver returns the time T_l until

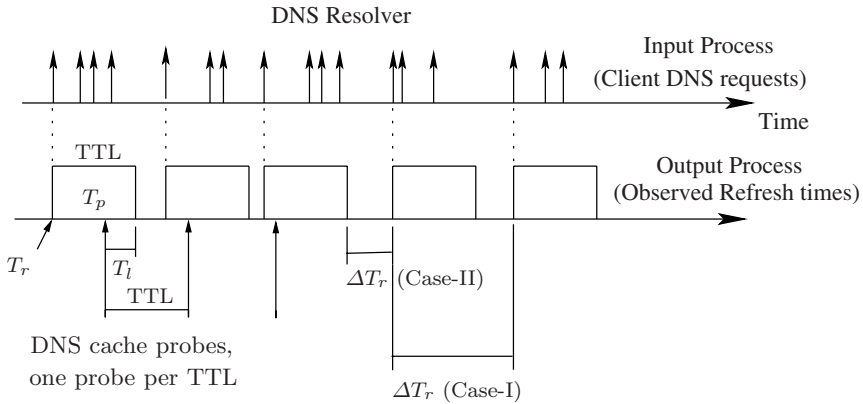


Fig. 1. Illustration of the estimation methodology

the cached entry expires. Given the TTL we can therefore infer the most recent start time (which we call the *refresh* time) T_r , as:

$$T_r = T_p - (TTL - T_i) \quad (1)$$

Then one way to estimate the average rate λ , is to compute the average time between consecutive refresh times $T_{r_1}, T_{r_2}, \dots, T_{r_R}$ (Case I in Figure [1](#)) from a sufficient number of refresh events R . Let ΔT_{r_i} be the time between consecutive refresh events of the entry S as observed from the output process, ($\Delta T_{r_i} = T_{r_i} - T_{r_{i-1}}$), then,

$$\lambda \approx \frac{R}{\sum_{i=1}^R \Delta T_{r_i}} \quad (2)$$

However, notice that this method is overly conservative since it assumes that no DNS queries arrive within the TTL of S in the resolver's cache. This is of course too restrictive and will lead to under-estimating the rate λ . Instead, we consider ΔT_{r_i} as the time between the expiry of the entry until its next refresh time (Case II in Figure [1](#)),

$$\Delta T_{r_i} = T_{r_i} - T_{r_{i-1}} - TTL \quad (3)$$

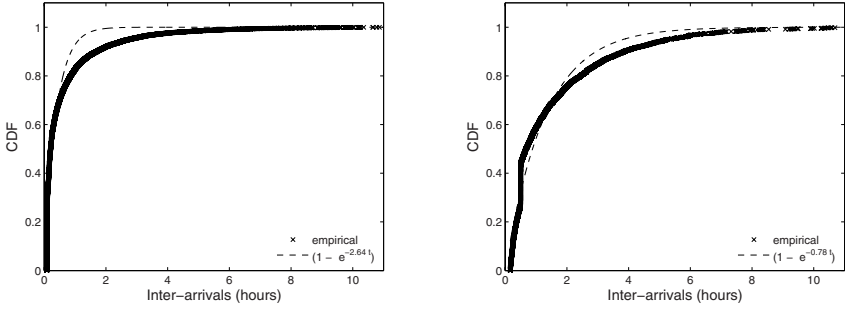
Notice that doing so makes the implicit assumption that the last DNS query in the input process took place slightly before the DNS entry expired. As we show later, this is not a significant issue and our technique still yields a fairly accurate estimate for practical TTL ranges.

Based on the newly calculated ΔT_{r_i} , we use Equation [2](#) to calculate the estimated rate λ from a sufficient number of refresh events R . To determine the required number of samples R , we apply the results of the central limit theorem [33](#). For an acceptable error e , and confidence $z_{\alpha/2}$, we can calculate the sample size accordingly (see Appendix [A](#) for the detailed derivation). Finally, with the estimated λ at hand, we can infer the number of clients as a function of λ and the individual client request rate λ_c . In the next section we discuss how we estimate both λ_c and n .

2.2 DNS Request Arrival Model

In order to estimate the number of clients n , we need some knowledge of the DNS request arrival model. We derive this model by studying the distribution of the inter-arrival times of incoming DNS requests to a particular resolver. Specifically, we use a large dataset of over 320 million NetFlow records collected at the edge of a large campus network during a 24-hour period on 7/15/2007. We use this dataset to study the arrival models for two popular domain names, namely, `www.google.com` and `www.cnn.com` with TTLs of 5 and 10 mins, respectively. Assuming that each HTTP connection is preceded by a DNS request, we deduce these models by extracting the inter-arrivals of the start times of flows originating from individual hosts and destined to each one of these domains. Jung *et al.* [21](#)

used a similar approach in their study on the effectiveness of DNS caching. Figure 2 shows the distribution of the inter-arrival times of requests to each name. As the graphs show, the incoming client DNS request arrivals can be reasonably modeled by exponential random variables with different rates λ_c (= 2.6 queries/hour for `www.google.com` and 0.78 queries/hour for `www.cnn.com`).



(a) `www.google.com` ($\lambda_c = 2.63$ queries/hour) (b) `www.cnn.com` ($\lambda_c = 0.78$ queries/hour)

Fig. 2. Cumulative Distribution Function (CDF) of the incoming client request inter-arrivals

Following the assumption from Section 2, the sequence of IID exponential inter-arrivals from n clients (each with an input rate of λ_c) generates an output arrival process with Gamma distributed arrival times $Gamma(n, \lambda)$. Since n in our case is an integer value, then it follows from the gamma distribution [33] that the output process has exponentially distributed inter-arrivals with an aggregate mean rate of $\lambda = n\lambda_c$. We use this property to indirectly estimate n from the measured output process rate λ , where λ is estimated using R refresh events as illustrated in Section 2. Given λ , the expected number of clients, is $\left(n = \frac{\lambda}{\lambda_c}\right)$.

3 Validation

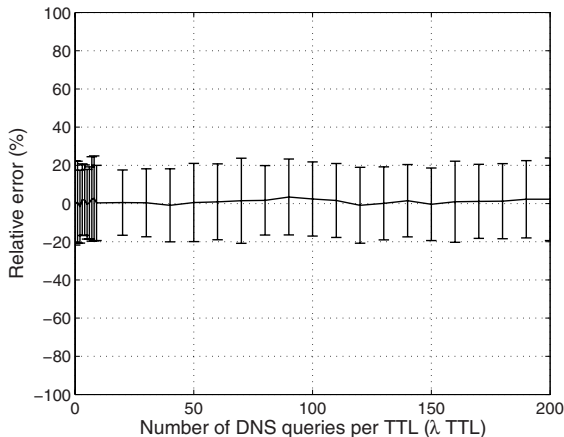
We first verify the accuracy of the proposed approach via simulation. Using the simulation parameters shown in Table 1, we evaluate the accuracy of our approach by measuring the estimation error of n for a wide combination of λ and TTL values. Figure 3 shows the estimation error. As the graph shows, our estimator is fairly accurate; the 95% confidence interval (i.e., within 2 standard deviations) of the mean estimation error remains within the bounds of the acceptable error set in the simulation.

3.1 In the Wild Evaluation

We further validate the effectiveness of our estimation technique by applying it to data collected from a real-world DNS probing experiment. In this experiment, we

Table 1. Simulation parameters

Acceptable estimation error (e)	20%
Actual number of clients	100
Confidence	95%
Number of samples (R)	100

**Fig. 3.** Relative estimation error of the number of hosts n under different number of DNS queries per TTL (λ TTL)

probe a local resolver serving a small department network for two popular DNS names, namely, `www.google.com` and `www.cnn.com`. For validation purposes, we enabled DNS logging on the local resolver so that all internal DNS queries issued for either one of those names were recorded. We then extract the unique sources making queries for each name and use their count as a validation baseline for our estimate. Table 2 shows the parameters used in our experiment as well as the estimation results. The client query arrival rate, λ_c , is chosen based on the campus-wide trace discussed in Section 2.2.

As the table shows, the estimates are fairly accurate. For example, the estimated aggregate rate λ for `www.google.com`, from our cache probing, is 240 queries/hour. Dividing this estimate by the client query rate, $\lambda_c = 2.63$ query/hour estimated from the NetFlow dataset, yields an estimate of 93 clients accessed `www.google.com`. Similarly, our evaluation yields an estimate of 30 clients accessed `www.cnn.com`. Both estimates are within the bounds of the 20% error margin set in the experiment parameters. These results provide evidence on the viability of this estimation technique. We now turn to illustrate the flexibility of our approach through two important security applications: web-metering and botnet size measurement. We believe these two applications serve as good examples of the strength of our scheme.

Table 2. DNS cache probing experiment, parameters and results

	www.google.com	www.cnn.com
TTL	300 seconds	600 seconds
Client mean query rate (λ_c)	2.63 query/hour	0.78 query/hour
Cache probing rate	1 query every 5 mins.	1 query every 10 mins
Number of samples (R)	100	100
Acceptable estimation error	20%	20%
Actual number of clients (n)	104	26
Estimated mean aggregate rate (λ)	240 queries/hour	23 queries/hour
Estimated number of clients (\hat{n})	93	30

4 Applications

4.1 Estimating Website Popularity

As mentioned earlier, web metering (or popularity ranking) plays an important economic role in today’s Internet. Popularity rank, for instance, is a key factor in deciding the marketing potential of a website. In particular, the higher a site’s popularity rank, the more advertisers are willing to bid for advertising space on that site. Not surprisingly, because of the strong correlation between website popularity and monetary benefits, techniques for rank inflation are not uncommon [12,12], and so this problem has stirred much interest on the design of secure metering schemes (e.g., [16,25]).

For the most part, web metering schemes attempt to address the problem of trust between advertisers and website owners by delegating the web metering task to a third party (e.g., Alexa [38], ComScore [20], NetRatings [26]) that monitors the interaction between clients and servers, and/or rely on cumbersome key agreement and distribution schemes [6,16,25]. In practice, the most well-known ranking services offer ranking for websites based on the number of visits they receive. These visits are measured from data collected from millions of users who willingly install measurement agents (e.g., Alexa toolbars [1]) on their machines. However, clearly such techniques raise security [18,34] and privacy concerns as they reveal user-specific traits to the ranking service. Furthermore, the resilience and accuracy of these techniques has been recently brought into question [1,22].

In what follows, we illustrate a simple, yet effective, web metering scheme that requires no client-side deployment. Additionally, our scheme is less intrusive as it does not breach individual users’ privacy. The outcome of the proposed scheme is a list of website ranks measured from a completely different perspective. These ranks can be used as a stand-alone measure of the relative popularity of websites or to validate the results obtained from other ranking schemes.

¹ Whether or not these toolbars should be classified as “*Spyware*” seems to be a subject of much debate lately.

Our technique is a direct application of our DNS-based estimation technique. That is, to measure the relative popularity rank for a set of websites, \mathcal{B} , we periodically probe the caches of a selected set of resolvers, \mathcal{D} , following the aforementioned approach (i.e., one probe every TTL epoch). Then to determine the relative rank of a website we simply measure the rate λ (as illustrated in Section 2) from the output of the probing process for each resolver in \mathcal{D} . The final rank K , is then expressed in terms of the average time it takes the web-site entry to be refreshed in the resolver cache after its last expiry (i.e., $1/\lambda$) across all resolvers in \mathcal{D} . Intuitively, DNS entries of websites with higher hit rates will be refreshed quicker than those with lower hit rates. To get the final website rank K we calculate the weighted average of the refresh times across all resolvers in \mathcal{D} ,

$$K = \sum_{i \in \mathcal{D}} \frac{W_i}{\lambda_i} \quad (4)$$

where, W_i is the relative weight of each resolver in the final rank outcome. A number of criteria can be used to decide the relative weight for each resolver. For example, the weights can be decided based on the population demographics and target market of the advertising company, or from light-weight sampling of the IP-space (e.g., [9,29]). In our case, we choose to apply a weight for each resolver based on the total client population served by that resolver. For simplicity, we infer this information from a dataset obtained from Google Inc. that contains a large list of resolvers and a coarse-grained estimate of the number of clients served by each resolver. For each resolver the weight W_i is then calculated as the number of clients served by resolver i divided by the total number of clients served by all resolvers in the sample \mathcal{D} .

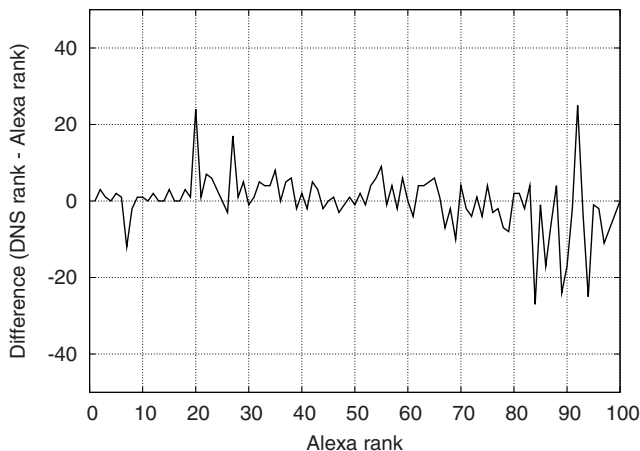


Fig. 4. DNS rank versus Alexa rank, for the top 100 websites according to Alexa ranking

Case Study. We apply the above methodology to measure the relative popularity of the top 100 web-sites according to the ranking of [Alexa.com](#)². For our target resolvers, we use a large list of 1.6 million resolvers obtained by collecting the Name Server (NS) records of a large list of crawled web URLs [27](#). The resolvers list is first sanitized to extract the “cooperative resolvers” (resolvers that respond correctly to external cache queries). The sanitization phase involves sending two consecutive DNS queries to each resolver for a known DNS record. The first is a recursive query that forces the DNS server to fully resolve the query. The second query is sent with the recursion flag turned off to elicit a local reply from the resolver’s cache. We compare the replies for consistency and also verify that the value of the *TTL* field in the second response is smaller than the one in the first response. After sanitization, a total of 768,000 resolvers were cooperative. As our target resolvers, we choose a smaller sample of resolvers from the sanitized list. We denote this sample \mathcal{D} . Notice that there are several ways to choose the sample \mathcal{D} . In our case, we first map the resolvers in the large list of 768,000 resolvers to their respective countries using the IP2Location database [13](#). Our resolvers mapped to 189 countries. From each country we randomly choose up to k ($=3$) resolvers to form our final target list \mathcal{D} of 495 resolvers [3](#). We choose this particular methodology to serve our goal of ranking website according to their popularity from a global perspective. However, the selection criteria can be tuned to serve other ranking goals. For example, one could select all resolvers from a certain country to study web-site popularity with respect to users from that region. Investigating the selection of resolvers to serve such goals is outside the scope of this paper.

We probe each resolver in \mathcal{D} for the top 100 websites from Alexa following the above methodology. Then we estimate λ_i for each resolver based on a sample of 50 probes and compute the final rank for each name using Equation [4](#). Figure [4](#) shows a comparison between our ranks and those of Alexa. As the graph shows, while both rankings show comparable results (with an average rank difference of 4.5), in some cases the ranks differ significantly. A closer look into some of the differing ranks reveal that they refer mostly to websites that have a country specific domain (e.g., [www.ebay.co.uk](#) had a rank difference of 22). Recall that we select our target resolvers from all-over the globe, hence this discrepancy is likely a consequence of the resolver selection criteria. In some other cases (e.g., [www.orkut.com](#) which has a rank difference of 12), the reason for the discrepancy in ranking is unclear. While it is difficult to argue for or against the accuracy of either ranking (without a true baseline) these results highlight the benefit of having metrics from different perspectives. This is important as the multiple measures can reveal inconsistencies in some ranks, and can be used further to produce new, and hopefully more robust, ranks based on a combination of different measures.

² We use the top 100 Alexa global ranks that are based on traffic statistics as of September, 2007.

³ For countries containing ≤ 3 resolvers we choose all the available resolvers.

Resilience to Fraudulent Inflation

Click fraud. Click fraud schemes [23] have the dual effect of inflating the number of “click-throughs” on the Ads posted on a website and increasing the popularity of the website as the number of hits increases. While click-fraud may directly affect the ranks produced by direct counting schemes (e.g., those of Alexa), its effect on our ranks is more limited. For one, only those clicks originating from hosts served by resolvers in our randomly chosen sample, \mathcal{D} , may influence the ranking. More importantly, to influence the outcome of the probing process, these clicks need to persist over a long period of time in order to significantly change the average refresh rate.

Direct manipulation attacks. In light of our technique, one might attempt to inflate the popularity of a website by polluting the caches of the resolvers that we probe. Cache pollution is possible if the resolver allows recursive external DNS queries. In this case, the attacker may send a sequence of synchronized DNS queries for the service name of interest—spaced by the *TTL*—so that the DNS entry is refreshed immediately after its expiry. Consequently, our probing process will falsely yield a high refresh rate for the target resolver. However, for this attack to be effective, the attacker must target enough resolvers from \mathcal{D} to influence the final website popularity rank. To mitigate such attacks, our sample of resolvers is selected at random, by region, and refreshed periodically, thereby making pollution attacks more difficult (though not infeasible) to perpetrate.

4.2 Estimating Botnet Size

Another compelling use of our technique is that of estimating the size of a botnet. Botnets are networks of compromised hosts, called bots, under the control of human operators, referred to as botmasters. Botnets are primarily used for various types of malicious activities, including denial of service attacks, click fraud [12], software piracy, and spam. While botnets have only recently attracted the attention of the research community, several works on the topic have already emerged ([10, 11, 19, 28, 31]). In particular, several studies have attempted to address the specific question of botnet size and the subtleties involved in size estimates. For example, Dagon *et al.* used DNS redirection to divert bot connections to a darknet in order to directly count the number of bots [11]. While effective, their approach requires coordination with DNS authorities in order to perform the redirection. Other studies used botnet infiltration to directly count the number of bot ID’s observed on the botnet command and control channel [17, 31]. Unfortunately, botmasters are increasingly suppressing bot information on the command and control channel, thereby hindering the effectiveness of these techniques.

As a remedy, we proposed [31] a technique for estimating botnet sizes using DNS cache probing. Similar to this work, botnet sizes were measured by probing the caches of a large set of DNS resolvers for the DNS name of the botnet server. The total number of DNS resolvers returning a cache hit for the name in question

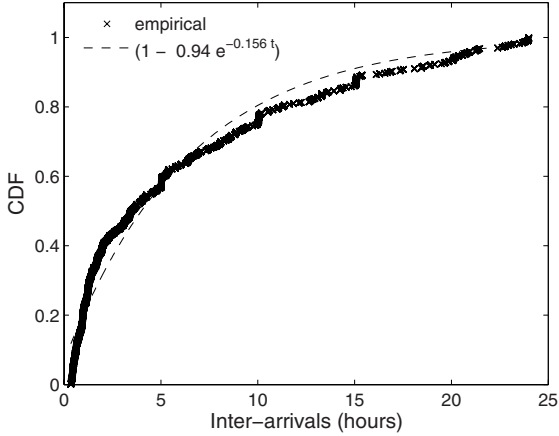


Fig. 5. CDF of Individual bot join inter-arrival times (based on data from 470 botnets)

provided a botnet’s so-called “DNS footprint”. However, this footprint is, *at best*, a lower bound of the true infection size because that approach does not provide any indication on how many infected bots reside behind a domain where a hit was returned.

In what follows, we illustrate how one can use the approach suggested in this paper to provide a better estimate of a botnet’s population. To do so, we first examine the distribution of bot request inter-arrivals. We derive this distribution by studying the bot⁴ join times extracted from a dataset containing the activity logs of 470 infiltrated IRC botnets [31]. Figure 5 shows the distribution of bot join inter-arrival times over a period of more than 9 months. As the graph shows, bot inter-arrival times can be approximated by an exponential distribution with an average rate of $\lambda_c = 0.156$ (i.e., an average of one connection every 6.4 hours).

Given knowledge of the distribution of bot request inter-arrivals, we now apply our estimation technique. In order to have a baseline for validation, we only choose botnets whose member counts are sufficiently large and can be calculated directly from the IRC traces. This yields two botnets, with member counts of 12,700 and 10,690 respectively. As our target resolvers, we use the large list of 768,000 cooperative resolvers from Section 4.1. Following the methodology from Section 2, for each resolver, we send a sequence of cache probes spaced by the *TTL* for the botnet server name in question then we estimate the botnet population by calculating the sum of the estimated number of bots n served by each resolver. Table 3 summarizes the parameters of the probing experiment and the results of our estimation.

⁴ In our analysis we assume that a bot IP address is a sufficient measure of bot uniqueness. To account for the effect of DHCP we only consider join inter-arrivals that are ≤ 24 hours. We also exclude bot joins resulting from clone attacks as well as any join with no associated quit message.

Table 3. DNS cache probing experiment, estimating botnet sizes

	Botnet I	Botnet II
TTL	15 mins	30 mins
Client mean query rate (λ)	0.156 query/hour	0.156 query/hour
Cache probing rate	1 query every 15 mins.	1 query every 30 mins.
Number of DNS resolvers	768,000	768,000
Number of samples (R) per resolver	100	100
Population size (from IRC logs)	12,700	10,690
Measured DNS footprint	1,700	1,452
Estimated population	8,400	6,350

The probing experiment shows that the botnets in question have DNS footprints [31] of 1,700 and 1,452, respectively. For each resolver in the footprint, we extracted all refresh times for both botnet server names. We then applied our estimator from Section 2 to derive the size of the infected population of each botnet. Our estimation results show that the sizes of the infected population were about 8,500 and 6,350 bots, respectively. Clearly, the population estimates derived from our analysis are much closer to the actual population sizes compared to the more coarse-grained DNS footprints — that imply sizes of only 1,700 and 1,452 bots, respectively. That is equivalent to more than a *three fold* improvement in accuracy over the DNS footprint estimate.

The observant reader would note that the error margin from the actual bot count is larger than that in Section 3. The degradation in accuracy is due to the fact that our list of target DNS resolvers only covers a subset of all DNS resolvers in the Internet. Hence, a more comprehensive list of servers would enhance the estimation accuracy. Additionally, botnet size instability (be it due to bot migration or churn [31]) also contributes to this effect. Nonetheless, we believe our result shows the utility of this estimation technique in assessing a botnet’s size when it is not possible to make such measurements directly even after the botnet has been infiltrated.

5 Practical Considerations

Notice that our probing mechanism requires cooperative resolvers that respond correctly to external cache probes. The sanitization step in Section 4.1 showed that roughly half of the resolvers in our list did not respond to external cache queries. While this is not a hindrance for some applications (e.g., for web-metering we only require a small sample of resolvers), having a large set of resolvers will improve the accuracy of other applications (e.g., botnet size estimation). An alternative probing model that can overcome this limitation would be to deploy a set of distributed DNS probing sensors inside the boundaries of large networks (e.g., within the boundaries of ISPs). The internal sensors will be able to query the caches of their respective resolvers and give an estimate for the client density in the same network.

Another noteworthy point of discussion is the practical impact of the *TTL* interval. Our analysis shows that the change in the *TTL* interval length does not significantly affect the accuracy of our estimator. However, in practice, the value of the *TTL* has an impact on the estimation speed and the overhead associated with the probing process. Recall that we probe each DNS name at a rate of one query per *TTL*. For large *TTL* values (e.g., on the order of one day), our probing scheme will require a long time to collect enough samples in order to reliably estimate λ . Luckily, major websites mostly use short *TTL* values for the purposes of load balancing [35]. Figure 6 illustrates the distribution of the *TTL* length for the top 100 websites in Alexa’s ranking. As the graph shows, the majority of the *TTL*s are relatively short (about 85% of the *TTL*s are less than one hour). Likewise, our earlier work showed that a significant portion of the DNS names used by botmasters have short *TTL*s [31]. In many cases, these DNS names are served by dynamic DNS providers that intentionally use shorter *TTL*s to accommodate for frequent IP address changes of the subscribing servers. Finally, we note that one way to accommodate for large *TTL*s is to compute a running estimate of λ and keep updating the estimate as more samples are collected.

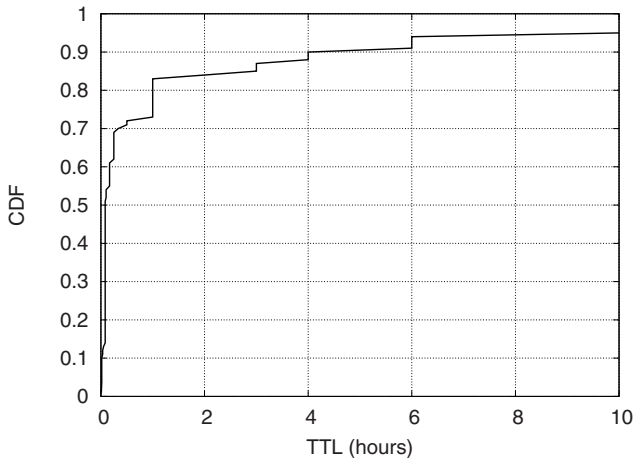


Fig. 6. CDF of the *TTL* intervals for the top 100 websites according to Alexa. The majority of websites use short *TTL*s with about 85% of the *TTL*s being less than one hour.

6 Other Related Work

The general problem of inferring the size of various client populations on the Internet has received considerable attention over the last few years. For the most part, the proposed techniques share the characteristic of attempting to estimate the size of a population in the absence of information from within the networks’

edge. Generally speaking, these techniques differ in the inference mechanisms they use to derive the different population estimates. Additionally, each scheme is normally tailored to meet a specific goal for a specific context.

For example, Bellovin proposed a technique for estimating the number of hosts behind a Network Address Translation (NAT) device [4]. His technique is based on observing the evolution of the value of the identity field in the outgoing IP datagrams. More recently, Casado *et al.* used the number of scans received by strategically placed darknets to infer the percentage of Code Red II-infected hosts that resided behind NAT devices [8]. Additionally, Casado and Friedman proposed techniques based on active web content to estimate the number of hosts located behind a large number of NAT devices and web proxy servers [7]. Our work is different both in scope and technique. In our case, the goal is to estimate the density of clients accessing the same Internet service using DNS cache probing.

The strength of our approach is demonstrated by its utility for a wide range of applications, two of which are presented in Section 4. The web-metering problem has been the subject of a number of earlier research proposals (*e.g.*, [6,16,25]). At a high level, these schemes use cryptographic primitives to design web-metering schemes that are resilient to click inflation attacks (*e.g.*, [1,2,12]). These approaches, however, are resource intensive and require sophisticated key agreement and key distribution schemes. By contrast, our scheme is relatively straightforward and requires no client side deployment.

Population estimation techniques have also been used to estimate the size of infections caused by malware spreading. For example, Dagon *et al.* used DNS redirection to measure the number of hosts connecting to IRC servers associated with botnet C&C channels [11]. More recently, we used both direct and indirect methods to better understand the spread of botnets in the wild, and how to characterize their behavior [30,31]. While these later works also use DNS probes, they are different from the approach in this paper in an important way: specifically, while our earlier work provides a course-grain estimate, we refine that approach to provide a technique for estimating (with reasonable approximation error) the number of infected hosts within these domains.

7 Conclusion

In this paper, we provide a new technique for estimating an important class of Internet demographics, specifically, the client population density of a given service. We demonstrate the utility of our approach through two applications that we argue are of much interest to the security and network community at large: verifying the popularity rank of a website and estimating the size of a botnet infection. Compared to earlier techniques, our popularity ranking scheme is easier to deploy, offers increased resilience to fraudulent manipulation, and is less intrusive as it does not reveal user-specific traits to the ranking service. In the second case, we provide a refined technique for estimating botnet size. We argue that since the issue of size plays an important role in assessing the monetary cost

and damage caused by botnets, improvements in accuracy in estimating their size is of immediate benefit. In short, our approach yields a three-fold improvement in accuracy compared to the best previously known technique. We believe these results aptly demonstrate the utility of our approach.

Acknowledgments

This work is supported in part by National Science Foundation grant CNS-0627611. We thank Angelos Keromytis, Sal Stolfo, Angelos Stavrou and Joel Rosenblatt for providing access to the anonymized NetFlow records used in our analysis. We also extend our gratitude to the reviewers for their insightful comments and feedback.

References

1. 20 Quick Ways to Increase Your Alexa Rank (2007), <http://www.doshdosh.com/20-quick-ways-to-increase-your-alexa-rank>
2. Anupam, V., Mayer, A., Nissim, K., Pinkas, B., Reiter, M.K.: On the security of pay-per-click and other web advertising schemes. In: WWW 1999: Proceeding of the eighth international conference on World Wide Web, pp. 1091–1100. Elsevier North-Holland, Inc., Amsterdam (1999)
3. Bailey, M., Cooke, E., Jahanian, F., Nazario, J., Watson, D.: Internet motion sensor: A distributed blackhole monitoring system. In: Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS) (2005)
4. Bellovin, S.M.: A Technique for Counting NATted Hosts. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement (IMW), pp. 267–272 (2002)
5. Bethencourt, J., Franklin, J., Vernon, M.: Mapping Internet Sensors with Probe Response Attacks. In: Proceedings of the 14th USENIX Security Symposium, August 2005, pp. 193–212 (2005)
6. Blundo, C., Cimato, S.: Sawm: a tool for secure and authenticated web metering. In: SEKE 2002: Proceedings of the 14th international conference on Software engineering and knowledge engineering, pp. 641–648. ACM Press, New York (2002)
7. Casado, M., Freedman, M.: Peering through the shroud: The effect of edge opacity on IP-based client authentication. In: Proceedings of 4th USENIX Symposium on Networked Systems Design and Implementation (NDSI) (April 2007)
8. Casado, M., Garfinkel, T., Cui, W., Paxson, V., Savage, S.: Opportunistic Measurement: Extracting Insight from Spurious Traffic. In: Proceedings of the 4th ACM Workshop on Hot Topics in Networks (HotNets-IV), College Park, MD (November 2005)
9. Chen, Z., Ji, C.: A Self-Learning Worm Using Importance Scanning. In: Proceedings of ACM Workshop On Rapid Malcode (WORM) (November 2005)
10. Cooke, E., Jahanian, F., McPherson, D.: The Zombie Roundup: Understanding, Detecting, and Disturbing Botnets. In: Proceedings of the first Workshop on Steps to Reducing Unwanted Traffic on the Internet (July 2005)
11. Dagon, D., Zou, C., Lee, W.: Modeling Botnet Propagation Using Time Zones. In: Proceedings of the 13th Network and Distributed System Security Symposium NDSS (February 2006)

12. Daswani, N., Stoppelman, M.: The Google Click Quality, and Security Teams. The Anatomy of Clickbot.A. In: Proceedings of the first USENIX workshop on hot topics in Botnets (HotBots 2007) (April 2007)
13. IP2Location Database, <http://www.ip2location.com/>
14. DNS Cache Snooping or Snooping the Cache for Fun and Profit, http://www.sysvalue.com/papers/DNS-Cache-Snooping/files/DNS_Cache_Snooping-1.1.pdf
15. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of internet miscreants. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 375–388. ACM Press, New York (2007)
16. Franklin, M.K., Malkhi, D.: Auditable metering with lightweight security. In: Financial Cryptography, pp. 151–160 (1997)
17. Freiling, F., Holz, T., Wicherski, G.: Botnet Tracking: Exploring a root-cause methodology to prevent denial-of-service attacks. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 319–335. Springer, Heidelberg (2005)
18. Exploiting the Google toolbar. GreyMagic Security Advisory, <http://www.greymagic.com/security/advisories/gm001-mc/>
19. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In: Proceedings of the 16th USENIX Security Symposium, pp. 167–182 (August 2007)
20. ComScore Inc., <http://www.comscore.com>
21. Jung, J., Burger, A., Balakrishnan, H.: Modeling TTL-based Internet Caches. In: Proceedings of IEEE INFOCOM (2003)
22. How Many Site Hits? Depends on Who’s Counting. New York Times article. Louis Story (2007), http://www.nytimes.com/2007/10/22/technology/22click.html?_r=3&pagewanted=1&ref=technology&oref=slogin
23. Metwally, A., Agrawal, D., Abbad, A.E., Zheng, Q.: On hit inflation techniques and detection in streams of web advertising networks. In: ICDCS 2007: Proceedings of the 27th International Conference on Distributed Computing Systems, Washington, DC, USA, p. 52. IEEE Computer Society, Los Alamitos (2007)
24. Moore, D.: Network Telescopes: Observing Small or Distant Security Events. In: 11th USENIX Security Symposium, Invited Talk (August 2002)
25. Naor, M., Pinkas, B.: Secure and efficient metering. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 576–591. Springer, Heidelberg (1998)
26. Nielsen/NetRatings, <http://www.nielsen-netrating.com>
27. Ntoulas, A., Cho, J., Olston, C.: What’s New on the Web? The Evolution of the Web from a Search Engine Perspective. In: Proceedings of the 13th International World Wide Web (WWW) Conference, pp. 1–12 (2004)
28. HoneyNet Project and Research Alliance. Know your enemy: Tracking Botnets (March 2005), <http://www.honeynet.org/papers/bots/>
29. Rajab, M.A., Monrose, F., Terzis, A.: Fast and Evasive Attacks: Highlighting the challenges ahead. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 206–225. Springer, Heidelberg (2006)
30. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: My Botnet is Bigger than Yours (Maybe, better than yours): Why Size estimates remain challenging. In: Proceedings of the first USENIX workshop on hot topics in Botnets (HotBots 2007) (April 2007)

31. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A Multifaceted Approach to Understanding the Botnet Phenomenon. In: Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC), pp. 41–52 (October 2006)
32. FBI Botnet Cyber Crime Report (June 2007), <http://www.fbi.gov/pressrel/pressrel07/botnet061307.htm>
33. Ross, S.M.: Introduction to Probability Models. Academic Press, London (1993)
34. Naraine, R.: Unpatched Google Toolbar Flaw Presents ID Theft Risk, <http://www.eweek.com/c/a/Security/Unpatched-Google-Toolbar-Flaw-Presents-ID-Theft-Risk/>
35. Shaikh, A., Tewari, R., Agrawal, M.: On the Effectiveness of DNS-based Server Selection. In: Proceedings of IEEE INFOCOM 2001, vol. 3, pp. 1801–1810 (2001)
36. Shinoda, Y., Ikai, K., Itoh, M.: Vulnerabilities of Passive Internet Threat Monitors. In: Proceedings of the 14th USENIX Security Symposium, August 2005, pp. 209–224 (2005)
37. FBI Computer Crime Survey (2006), http://www.fbi.gov/page2/jan06/computer_crime_survey011806.htm
38. Alexa: the web information company, <http://www.alexa.com>

Appendices

A Deriving the Required Number of Refresh Events (R)

It is known that the average rate λ measured from multiple independent samples is normally distributed around the actual mean:

$$f(\lambda) = N\left(\lambda, \frac{\lambda}{R}\right). \quad (5)$$

For an acceptable estimation error e and a confidence $z_{\alpha/2}$ the number of samples R required is [33]:

$$R = \left(\frac{z_{\alpha/2} \sigma}{e}\right)^2. \quad (6)$$

where σ is the standard deviation of the measured mean rate, λ , which can be determined from a smaller pilot sample of DNS refresh events.

Pushback for Overlay Networks: Protecting Against Malicious Insiders[★]

Angelos Stavrou¹, Michael E. Locasto², and Angelos D. Keromytis³

¹ Computer Science Department, George Mason University

² Institute for Security Technology Studies, Dartmouth College

³ Computer Science Department, Columbia University

Abstract. Peer-to-Peer (P2P) overlay networks are a flexible way of creating decentralized services. Although resilient to external Denial of Service attacks, overlay networks can be rendered inoperable by simple flooding attacks generated from insider nodes.

In this paper, we study detection and containment mechanisms against insider Denial of Service (DoS) attacks for overlay networks. To counter such attacks, we introduce novel mechanisms for protecting overlay networks that exhibit well defined properties due to their *structure* against non-conforming (abnormal) behavior of participating nodes. We use a lightweight distributed detection mechanism that exploits inherent structural invariants of DHTs to ferret out anomalous flow behavior.

We evaluate our mechanism's ability to detect attackers using our prototype implementation on web traces from IRCache served by a DHT network. Our results show that our system can detect a simple attacker whose attack traffic deviates by as little as 5% from average traffic. We also demonstrate the resiliency of our mechanism against coordinated distributed flooding attacks that involve up to 15% of overlay nodes. In addition, we verify that our detection algorithms work well, producing a low false positive rate ($< 2\%$) when used in a system that serves normal web traffic.

1 Introduction

Peer-to-Peer (P2P) overlay networks are a powerful and flexible way of creating decentralized routing services for various applications, including content distribution and multimedia streaming [11][13][3][20], network storage [7][18][9], resilience [2], packet delivery using rendezvous-based communications [19] and denial of service (DoS) protection [12]. A large number of overlay networks, such as CHORD [6], CAN [16], PASTRY [17] and TAPESTRY [8], are *structured*; that is, they use Distributed Hash Tables (DHTs) to perform directed routing. The use of DHTs imposes an inherent structure which dictates a well-defined and bounded set of neighbors in each P2P node. These neighbors are used by the P2P node to communicate all of its requests and replies. In

[★] This work was supported by the National Science Foundation under NSF grant CNS-07-14277. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

addition, requests arriving from neighbors are appropriately forwarded to other neighbors until they are routed to the right overlay node. In a healthy overlay network, we expect that the load generated or forwarded by a node is, on the average, statistically similar (but not identical) to the traffic generated by any other overlay node; that is, we do not expect that the traffic will start deviating too much from the average traffic generated by other nodes in the system. This is even more true for overlay networks that provide a service where clients do not actively participate in the overlay, such as I3 [19], Oceanstore [9], SOS [12] and others.

In this paper, we investigate methods to identify the traffic anomalies that can arise from deliberate attacks. Our detection algorithms *do not depend on object popularity*. Instead, we base them on measures of aggregate packet flows. Doing so enables us to avoid exorbitant storage and processing costs and maintain scalability in terms of the number of overlay participants. Although other researchers have used statistical methods to detect and examine aggregate flows [5,14], we are the first to consider the use of such methods in an overlay network setting that takes into consideration the neighbor-structure of P2P systems.

In general, we can protect an overlay network that has the following properties:

- The neighbors of each overlay participant are known for a window of time
- We can determine (within bounds) the fraction of requests we expect to receive from each neighbor

The above properties hold for almost all DHT-based Overlays (CAN is an exception) and even some randomized ones where the set of neighbors is of fixed size and the search requests have a predefined maximum length. Although we do not address misrouting directly as in [4], we do not allow mis-forwarding or neighbor spoofing: every node has a fixed list of known authenticated neighbors using pair-wise symmetric keys. Only these neighbors are allowed to route packets through the node and only to valid destinations according to the structure of the overlay. All other traffic is dropped, making objects reachable only by nodes that follow proper routing. By exposing traffic anomalies, our work prevents nodes from dropping or injecting requests, thereby encouraging nodes to conform to “normal” forwarding behavior with respect to the rest of the flow aggregate.

1.1 Pushback-Like Protocol

Pushback [10] is a router-based mechanism for defending against DDoS attacks. In a Pushback-enabled routing system, a router cognizant of the bandwidth limitations of *downstream* nodes may adopt a more proactive forwarding strategy. Instead of sending packets down a congested link (where they would be lost) or dropping such packets itself (which does not address the root of the problem: too much inbound traffic), the router would instruct (some of) its upstream routers not to forward certain packets. Heuristics are employed to identify the flows, or “aggregates” (packets having some common property, such as the same destination IP address and TCP port), responsible for the downstream congestion. The router examines its incoming (with respect to those flows) links, and the fraction of upstream routers responsible for most of the incoming

packets belonging to the aberrant flow are asked to rate limit that flow. These routers, in turn, recursively apply this mechanism.

Pushback, as proposed [10], works best when malicious traffic is anisotropically distributed around the Internet, so that some routers might rate limit traffic more severely than others. It also requires some level of trust between routers, an expectation that turns out to be somewhat unrealistic when crossing administrative boundaries (*e.g.*, an ISP's border or peering routers). The Pushback system also assumes that the participating routers would not misbehave in their execution of the Pushback mechanism and protocol.

There are two fundamental differences between that original setup and an overlay network: in the latter, the nodes themselves are both the originators of traffic and overlay "routers", and we have to assume that some of them will be compromised. Furthermore, not only can a compromised overlay node flood the network with traffic but it can also drop traffic meant to be routed to another destination via one of its neighbors.

Thus, in an overlay network, we have two types of misbehaving flows: flooding flows and packet drops. For excessive flows we first rate limit the offending aggregate flow and notify the upstream overlay neighbor of the problem, expecting him to rate limit the offending flow. For packet drops, we contact our neighbors' neighbors asking them to give us the counts for the aggregate flows in question, until we find a conflicting count indicating the node that drops the packets or lies about its aggregate packet count. By recursive and distributed application, this pushback-like protocol allows us to isolate the attacking nodes, quenching at the same time the effects of the attack. Of course, if our system has a lot of attackers collaborating with each other to both not comply but also to falsify their aggregate rates, pushback itself can be exploited by the attackers to generate additional traffic to the overlay network. On the other hand, even if the attackers are a significant portion of the network, if they are not coordinated they can be identified and isolated by the rest of the overlay nodes. An important assumption is that neighboring nodes cannot fake their identities, that is, they cannot pretend to be another node in the overlay. There are many mechanisms that we can use depending on the underlying network and operating system cryptographic facilities available to the nodes. For example, if the underlying network is the public Internet, we can set up pairwise-authenticated encrypted tunnels using IPsec, GRE, or some other encapsulation protocol used to identify the origin of the traffic.

1.2 Our Approach

We test our implementation using traffic from actual web server caches [11] to drive the detection process. To that end, we extract the source of the request and the requested object and use their hash values to map them into sources and objects in our system. This way we test the performance our system under normal traffic tuning the necessary parameters to limit false positives. Our results show that:

1. We can efficiently detect and mark excessive flows even when up to 25% of the system's nodes have been compromised.
2. Our algorithms require $O(\log^2(N))$ of memory per node, where N is the overlay nodes.

3. The proposed pushback-like protocol remains effective even when up to 15% of the overlay nodes attempt a coordinated attack

Our work is the first that attempts to detect, identify and isolate DOS flooding attacks initiated from inside an Overlay Network. The novelty of our approach lies in the exploitation of the properties inherent in these P2P systems with inference-based techniques.

2 Flow Model Description

This Section presents the notation we use to describe what we consider a “structured” P2P system, formally defines our notion of an attack, and provides a description of invariants that structured P2P systems exhibit.

2.1 Structured P2P Systems

A structured P2P system maps a set of keys K_{ids} to a set of nodes of size N and provides a distributed routing algorithm among these nodes. When a node n wishes to forward a message to the node holding key k , each P2P node on the route forwards the packet to the next P2P node along the path.

A *flow*, (s, k) , consists of the set of search requests sent from node s to the key $k \in K_{ids}$. Let $\lambda_{S,K}$ be the rate of packet transmissions from nodes in S to keys in K , and let $\lambda_{avg}^K = \frac{\lambda_{S,K}}{|S|}$, *i.e.*, it is the average rate at which a node transmits requests toward keys in K . Objects stored in the P2P system may have different popularity, *i.e.*, that in general $\lambda_{avg}^{k_1} \neq \lambda_{avg}^{k_2}$ for $k_1 \neq k_2$.

Initially, we also assume that for a fixed object k , the popularity of this object is similar among the participants of the P2P network: $\lambda_{S,k} = \lambda_{avg}^k$ for all S, k . We expect that as $|S|$ grows, if the nodes that comprise S are chosen at random, then $\lambda_{S,k}$ will quickly approach λ_{avg}^k .

2.2 DoS Attackers and Attack Intensity

We consider DoS attacks targeted toward a specific key or set of keys. Such an attack could be mounted to block access to data associated with a particular key. A node that is the origin point of excessive packets toward key k is said to be an *attacker* of key k .

In a healthy P2P network, the rate of a flow $\lambda_{S,k}$ from a fairly large, randomly selected group of nodes S toward a set of keys K should closely approximate the popularity of that object λ_{avg}^K . We say that a flow aggregate (S, K) is *misbehaving* if $\lambda_{S,K} > (\delta + 1) \cdot \lambda_{avg}^K = \lambda_{max}^K$ for some $\delta > 0$, where δ represents a lower bound on the proportional increase that a flow can transmit relative to the average rate before the flow is labeled as misbehaving. The previous notion can be extended to a set of nodes S as $\lambda_{S,K} > |S| \cdot (\delta + 1) \cdot \lambda_{avg}^K = |S| \cdot \lambda_{max}^K$. Note that for a set of nodes, the maximum rate allowed before we declare the aggregate flow as misbehaving depends on the size of the set. The selection of δ is a measure of the tolerance that we allow between different nodes (or groups of nodes) in the P2P system before declaring that a flow is misbehaving. If we assume a totally homogeneous system, then $\delta = 0$ — *i.e.*, no deviations from

the average rate (popularity) are allowed. Larger values allow more tolerance, but also give an attacker more freedom to deviate from the average rate and avoid detection. Typically, we select $\delta = 0.1$ which means that we detect flows that send ten percent more than the average rate of a set of keys K .

We define β to be the proportion by which one attacker increases its traffic toward k above λ_{avg}^k such that the attacker transmits packets toward key k at a rate of $(\beta + 1) \cdot \lambda_{avg}^k$. If N_a is the number of attacking nodes, the total amount of excessive search requests injected into the system by the attackers is $\beta \cdot N_a \cdot \lambda_{avg}^k$. The rate of search requests λ_{attc} , the target node experiences under attack is:

$$\lambda_{attc} = \frac{N_a}{N} \cdot (\beta + 1) \cdot \lambda_{avg}^k + \frac{(N - N_a)}{N} \cdot \lambda_{avg}^k$$

Let $f = N_a/N$ be the fraction of nodes compromised; we define the attack intensity, or gain due to the attack, D_A , to be the increase in the popularity of the target key k caused by the attackers' excessive search requests:

$$D_A = \frac{\lambda_{attc}^k}{\lambda_{avg}^k}$$

For example, if $D_A = 2$, the node that stores the object under attack has to serve twice as many search requests for that object. For the attackers' queries to be harmful to the target node that stores the keys being attacked, D_A must be large. If an attacker only controls a limited number of nodes, their only choice is to increase β . A large β and small f means that there will be a relatively small number of attack flows, and that these attack flows will inject significantly more traffic toward k . Our methods to detect misbehaving nodes will utilize the following measure:

- Fixed-Key Variable-Source (FKVS):

$$\alpha(c, K, S_1, S_2) = \frac{\lambda_{S_1, K}^c}{\lambda_{S_2, K}^c}$$

The FKVS measure compares the rates of two sets of sources for a particular set of keys. In a healthy P2P system, for appropriately sized (large) sets S_1 and S_2 , if c lies on the paths from all S_1 and S_2 to the nodes storing keys K , we should have that $\alpha(c, K, S_1, S_2) = |S_1|/|S_2|$.

3 Statistical Bounds of Flows

We now present methods that use the previously introduced metrics to identify misbehaving flow aggregates and mark packets that belong to these aggregates. The total number of possible flows in a P2P system is $N \cdot |K_{ids}|$. Thus, tracking each individual flow would require $O(N \cdot |K_{ids}|)$ memory. Even if we fully distributed the tracking load amongst all participating nodes, $O(|K_{ids}|)$ memory would be needed to collectively track all flows.

To avoid utilizing such a potentially huge amount of memory per node, we track a set of aggregate flows whose size is $O(\log(N))$ by taking advantage of the fact that the number of flows in each of the incoming and outgoing neighbors is $O(\log(N))$. We require that each node c consider as a separate aggregate all flows that arrive via the same incoming neighbor and exit via the same outgoing neighbor. Hence, there are $O(\log^2(N))$ aggregates to consider. By numbering the neighbor nodes in In_c and in Out_c , we can identify the flow aggregate that enters through the i -th neighbor and departs through the j -th neighbor as $f_{i,j}$, where $i = 0$ implies the flow originates at c and $j = 0$ implies the flow terminates at c . Aggregate $f_{i,j}$ is assigned a counter, $C_{i,j}$. When a packet arrives, we increment the counter that corresponds to the aggregate flow to which the packet belongs.

3.1 Comparison of Aggregate Flows

Our detection algorithm compares each flow's counter $C_{i,j}$ to the counter for all the aggregate of flows that exit to the same outgoing neighbor. We wish to determine the likelihood that, in a healthy P2P system, $C_{i,j}$ can have the value observed, under the assumption that C_j is made up mainly of healthy flows. More formally, let $X_{i,j}$ and X_j respectively be random variables that equal the value of these counters in healthy P2P system, and determine $\mathbb{P}(X_{i,j} \geq C_{i,j} | X_j = C_j)$. (Note that $C_{i,j}$ and C_j represent actual observed values in the real system, whereas $X_{i,j}$ and X_j are values that occur in a trial on top of a healthy P2P system.) The larger this probability, the more confidence we have that flows entering through neighbor i and exiting out of neighbor j are healthy. We define ϵ to be our *level of confidence* of the test. If $\mathbb{P}(X_{i,j} \geq C_{i,j} | X_j = C_j) < \epsilon$, then (according to this test) there is a probability $< \epsilon$ that $f_{i,j}$ is healthy. By comparing this value to the observed values of the counters, we can determine, within a certain confidence level, whether one flow's rate is higher with respect to the all the other flows' rate.

In the previous calculations, we don't need to assume anything about the flows' distribution. While any slight deviation from the normal transmission rate could be flagged as a violation, we allow a small degree of variability. Hence, we apply our "slack" factor, δ , and say that flow $f_{i,j}$ is misbehaving when the actual ratio of observed rates is larger than $(1 + \delta)\alpha(c, K, S_i, S'_i)$ for the second test. Setting δ to 0 leaves no slack. If $f_{i,j}$ sends at a rate even slightly above its supposed rate in a healthy P2P system, the central limit theorem tells us that eventually, $f_{i,j}$ will be flagged as unhealthy. Setting δ to larger values allows for additional slack.

4 Application to a DHT System

A Chord peer to peer system consists of a set nodes of size N that try to serve objects that are hashed and stored in nodes using an m bit hash function. The key identifiers K_{ids} are placed in circular order, creating a ring of length $K = \|K_{ids}\| = 2^m$. To simplify the notation, all math in the remainder of this section is performed modulo K . Each node is assigned an identifier (id) from the key space, thus creating a node ring. Since we have a circular placement, we have for each node c a successor node and a predecessor node. Each node is assigned a set of keys, meaning that the node either stores or knows the location of all the objects in its local database that hash to its value.

4.1 Chord Invariants and Tests

Proposition 1 *Let $i_1 > i_2$. If the set of flows (S_{i_2}, K) that pass through c is non-empty, then:*

$$\alpha(c, K, S_{i_1}, S_{i_2}) = 2^{i_1 - i_2}.$$

Proof. In the Appendix

Proposition 2 *If the set of flows (S, K) that pass through c is non-empty, where $S = \cup_j S_j$, then:*

$$\alpha(c, K, S_{i_1}, S) = \frac{1}{2^{\log(N) - i_1} - 1}$$

Proof. The proof follows the form of Proposition 1, noting that there are $\sum_{j=0}^{\log(N) - i_1} 2^{i_1 - j}$ sources from S to K that enter c (through any finger).

5 Experimental Results

5.1 Web Trace-Driven Simulations

To test our implementation, we used web traces obtained from IRCache repository [11] to drive our simulated environment. As we will soon present, our experiments show that under non-attacking conditions our system does not generate excessive false positives (less than 2% false positives).

Our first goal was to verify the effectiveness of our detection algorithm and to evaluate its performance. To that end, we use an implementation of the Chord peer to peer network in a series of simulations. In all of our experiments, some of the participating peer to peer nodes assume the role of the “attacker”. The “attackers” select a key at random from the set of allowed keys and generate a disproportionate number of search requests towards that key. The node that stores the key under attack is the target. The goal of the attackers is to flood the target with search requests, crippling its ability to respond. In general, the attackers are allowed to select multiple targets simultaneously. However, if we assume that attackers have abundant but nonetheless limited resources, aiming at multiple targets will only lower their aggregate attack ability. Indeed, the attackers will have to split their attack requests between the different targets, reducing their attack intensity. We formally defined attack intensity as the increase in the search request traffic towards the target key caused by the attackers’ excessive search requests. For example, an attack intensity of $D_A = 2$ means that the node that stores the key under attack has to serve twice as many search requests for that object as it would normally have.

We use the notion of attack intensity to formally quantify the attack ability of the compromised nodes. Where our test is working perfectly, marking only excessive search requests, only a fraction $\frac{\beta}{\beta+1}$ of packets in the attacking flow should be marked. We determine our performance by measuring our ability to detect the attack as close to the attacker as possible. Also, we want to mark the malicious search requests as many times as possible along the path from the attacker to the target object. All of these metrics are dependent both on the attack intensity and the relative attacker’s distance from attacked key.

Another goal of our experiments was to identify the limits of our detection algorithm. Our attack detection method identifies aggregate groups containing attackers by marking packets from these groups as “excessive” whenever our estimates predict that the groups are transmitting at too high a rate. Since we operate at the granularity of groups, we introduce a false positive error for the non-attacking individual flows which happen to be grouped with “excessive” ones. As we show, this error is relatively small because the vast majority of the requests from a malicious aggregate flow belong to the attacker. Moreover, the attack requests get marked multiple times along the path from the attacker to the target. Another potential source of error can arise due to the use of statistical inference.

Of course, this error can become arbitrarily small by selecting a higher confidence interval. In our simulations we used a 0.999% confidence interval.

In the last set of experiments, we use a pushback-like protocol where a node, upon detection of a misbehaving aggregate flow, communicates with its upstream neighbor that this flow is originated. The misbehaving aggregate flow, if it is excessive, is rate limited to the average of the rest of the flows. If the upstream neighbor fails to respond with a certain amount of packets, which is a parameter for our system, this node is considered malicious and the rest of the overlay nodes are notified. The protocol is applied recursively until the source of the anomaly is identified or the flow stops being excessive. Either way the DoS attack will be prevented allowing only small, negligible spikes of packets to reach the target node.

To ensure the statistical validity of our experiments, we used approximately 4 million search requests per simulation. In addition, each simulated experiment was repeated more than 50 times. The results we present in our graphs are the average of these simulations; the variance among the different experiments was observed to be low.

5.2 Detection of Single-Attacker

We start by examining the scenario where a single node is compromised. Although simplistic for a real world attack, this scenario provides insights on the effectiveness of our approach. Furthermore, we can evaluate our ability to detect the malicious node for varying distances relative to the target and for a range of attack intensities. Initially, we placed the attacker in various distances (in hops) away from the node responsible for the target key, the target node. We then measured the percentage of the excessive search requests our algorithm detected and their detection distance from the target *i.e.*, how quickly was the traffic identified as excessive. As the distance of the attacker from the target increases, the detection distance increases accordingly. In addition, we detect

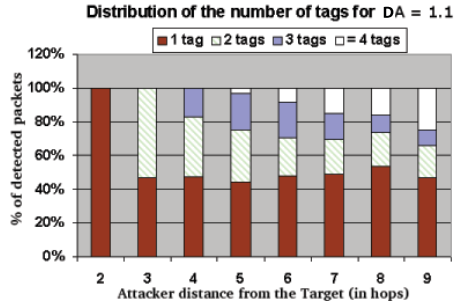


Fig. 1. Distribution of the number of tags for the attack requests for one attacker in a 1024-node Chord ring. The different plots represent the attacker’s distance in hops from the target for attack intensity of 110% ($D_A = 1.1$) *i.e.* the object receives 10% more traffic.

the malicious flow in multiple nodes along the path from the attacker to the target. It appears that the largest portion of the attack requests are detected close to the attacker. For example, if we place the attacker at a distance 9, the majority of its packets are detected by its next hop neighbor, with distance 8, then by the next node at distance 7 etc. Even when the attacker’s proximity to the target is reduced to the minimum, *e.g.*, at a distance of 2, we detect 100% of the excessive search requests at the immediate neighbor.

Recall that each node tags all the flows of a group that appears to be “misbehaving”. Figure 2 shows the relative distribution of the tags when we vary the distance of the attacker to the target. We see that, depending on the distance between the attacker and the target, a significant portion of the excessive search requests are tagged at least twice by nodes along the path from the attacker to the target. The number of tags increases as we increase the number of hops between the attacker and the target since the attack packets traverse more nodes in order to reach their target.

To actually have impact on the search requests of the attacked object, the attacker’s intensity, D_A should be relatively high. For example, for $D_A = 2$ the single attacker has to inject search requests in the system with rate $N \cdot \lambda_{avg}^k$, or with a $\beta = N$. Although this rate may appear unnecessarily high, in practice this depends on the popularity λ_{avg}^k of the object attacked. If the object is highly popular and λ_{avg}^k is large compared to the rest of the objects in the system, the attacker will need to significantly increase the number of search requests to noticeably affect its popularity.

We have shown that our method detects and marks search requests on groups of flows. An inherent problem is that we may end up marking legitimate flows along with the attacking ones (since they are mixed in the same aggregate) but that is not the case for our method. Although we are marking flows belonging to the same “misbehaving” group, we are punishing mostly the attacker since it is the one sending the majority of the search requests through that group. The majority of the other flows are getting marked minimally, in comparison both to the total number of requests they generate and to the attacker requests marked. Naturally, blindly marking and dropping excessive requests from a misbehaving flow is a very crude method to prevent a DoS attack, although it can be effective if resources are otherwise limited.

Through our experiments we wanted to ensure that there is no attacker placement inside the Chord ring that our algorithm fails to detect. We are now in position to present more realistic results from simulations in which we have one attacker randomly placed

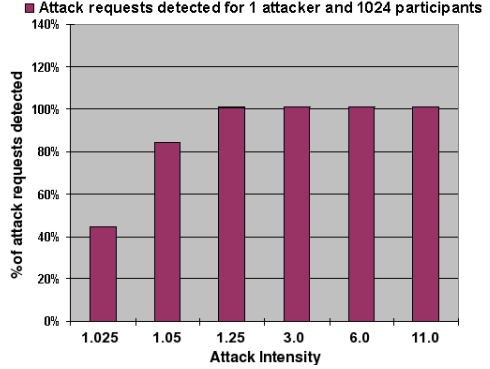


Fig. 2. Attack packets detected for one attacker randomly placed in a 1024-node Chord ring. The different bars correspond to increasing values of attack intensity D_A . The results presented are the average of multiple experiments (100 per bar).

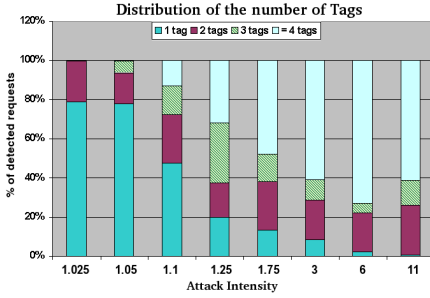


Fig. 3. Distribution of the number of tags for the detected attack packets for a 1024-node Chord ring with randomly selected attacker-target placement. Each value on the X axis corresponds to exponentially increasing attack intensity (D_A). The results averaged over 100 experiments for each attack intensity value.

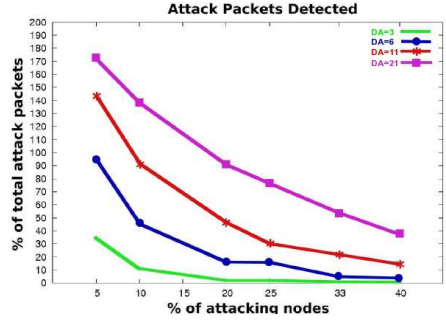


Fig. 4. Percentage of attack packets detected when we vary both the attack intensity D_A and the fraction of nodes compromised for a 4096-node Chord ring. Each line represents different attack intensity values. We can see that as we increase the attack intensity we can detect more percentage of the attackers' request even when the attack is very distributed.

in a Chord ring of size 1024 (see Figure 2). We observe that even for very low attack intensities values, $D_A = 1.025$, a mere 2.5% increase in the load of the end server, we can detect more than 40% of the attack packets. It is easy to see that for attack intensity values larger than 0.05 our method detects a significant portion of the excessive requests. As the intensity of the attack diminishes, our detection results become weaker. This is something we expected, since our algorithm detects excessive requests based on measurements done on groups of flows where small variations in the intensity of one flow does not have significant impact on the aggregate flow. For some values of attack intensity, especially for $D_A = 0.25$, we have an over-marking of the attacker search requests which fades out when the attack intensity becomes more significant. This is due to the group detection nature of our algorithm and the fact that aggregate groups of flows contain both legitimate and excessive flows originating from the attacker.

The number of tags for the attack packets is an increasing function of both the average attacker distance and of the attack intensity as shown in Figure 3. The average attacker distance seems to play a more prevalent role. This means that as the average distance between the attacker and the target increases, so does our ability to tag the attacker on multiple locations along the attacker-target path. Thus our system works better as we increase the number of participants in the DHT system, since the average distance between two nodes in the system increases. The detection behavior of our algorithm for attack intensities that are higher than $D_A = 11$ is similar to those measured for $D_A = 11$ and we omit them from our figures.

5.3 Detection of Multiple Attackers

We now study the behavior of our detection algorithm using a Chord ring where we vary both the fraction of nodes compromised and the attack intensity. Figure 4 presents the results for a 4096-node Chord ring with multiple attackers. It is clearly shown that there is a correlation between the excessive search requests detected and the attack intensity. Our results show that as the attack becomes more severe, our ability to detect excessive search requests increases; even when 40% percent of our nodes are compromised we are able to detect around 50% of the excess requests.

On the other hand, as the proportion of compromised nodes grows, there is a corresponding drop in our ability to detect excessive search requests, since the attack becomes more distributed on the Chord ring. Additionally the number of tags for the excessive requests are inversely proportional to the fraction of nodes compromised. For example, as Figure 5 shows, when the attackers constitute 5% of all the Chord participants, a large portion of their excessive requests have 2 or more tags, whereas when the attackers become 40% of the total, only 14% of the excessive requests have 2 or more tags.

In some cases we overestimate the number of attack packets sent by the attackers. This happens because we detect groups of flows where, on average, the attacker participates with multiple flows, leading to over-marking of search requests generated from the attacker. This over-marking can be used to weed out the specific flow from the group of flows detected to be misbehaving.

Another source for this overmarking comes from the fact that we allowed the non-attacking nodes to select the popularity of each key/object from a uniform (0, 1] distribution. This generates different preference distributions for each node pushing our “normality” assumptions to their limit. However, it also confirms our initial assumption that our analysis remains the same even if we allow different preference distributions. With a mean of 1/2 and a standard deviation of 1/6, the uniform distribution allows for wide spreading of the possible weight values in the full (0, 1] spectrum. The use of a normal or exponential distribution for the same interval would have resulted in less variance, leading to a more concentrated weight distribution. The law of large numbers dictates that any type of weight distribution would eventually lead to a normal preference distribution on the limit as the number of keys grows.

In Figure 6 we can see the percentage of packets detected for a Chord ring of size 1024. Again, each line represents different attack intensities. The detection ability dissipates as the percentage of attackers in the total node population increases. Note that the

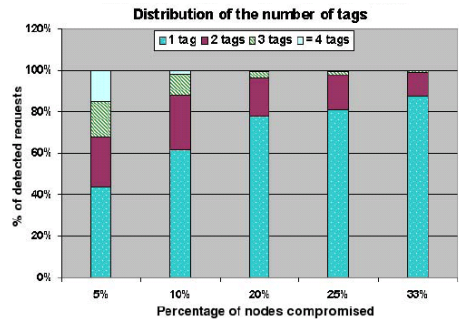


Fig. 5. Distribution of the number of tags for the excessive search requests detected when the attack intensity is $D_A = 21$ and we vary the fraction of nodes compromised for a 4096-node Chord ring. Notice that the number of tags on the attack requests are inversely proportional to the fraction of the attackers. The more “distributed” the attack is, the more difficult it is to detect and tag.

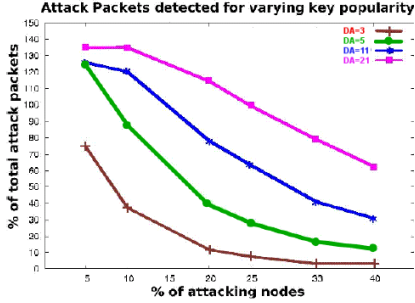


Fig. 6. Percentage of excessive packets detected when we vary both the attack intensity D_A and the fraction of nodes compromised for a 1024-node Chord ring. Each line represents different attack intensity values. For this experiment we allowed each individual non-attacking node to assign a weight to each key selected from a uniform distribution. The detection results appear to be better than using the same preference distribution function.

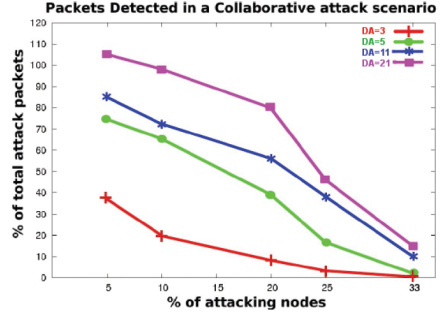


Fig. 7. In this experiment, the attackers are collaborating by not marking the excessive packets destined for the target key only (co-ordinated attack). Each line represents different attack intensity values and we vary the percentage of nodes compromised. Even under a collaborative attack, and with a significant portion of all nodes being compromised, we detect a large portion of the attack requests.

detection results are better than those presented in Figure 4, where we used the same preference distribution function for all the nodes. Indeed, when using different preference distribution, sometimes attackers are grouped with flows that have high preference for the attacked key and thus revealed faster. However, for the same reasons, we end up having a higher false positive detection of around 0.1% compared to the 0.07% when we used the same preference function. Overall, allowing the nodes to have a different preference distribution function leads to better detection results at the cost of a slight increase in the false positive detection percentage.

In our experiments, we used a fixed value for the threshold δ , namely $\delta = 0.1$. Recall that δ is a parameter of our detection algorithm, indicating our tolerance towards deviations from the average popularity of a key. Larger values allow more tolerance and lower our false positives, but also give an attacker more room to deviate from the average rate before our algorithm starts detecting the excessive requests. There is a trade-off between speed of detection and false positives. Figure 10 shows that, for $D_A = 20$, we get an improvement in our detection as we decrease δ from 0.1 to 0.05. At the same time we see an increase in the false positive percentage, which becomes more apparent for $\delta = 0.05$. Conversely, for $\delta = 0.07$ we get an improvement of almost 12% in the detection rate, when we have a distributed attack with $> 20\%$ of attackers. Lowering δ below 0.07 does not improve the detection rate, doubling false positives.

Our last experiment stresses our detection algorithm under the worst-case scenario: a highly distributed attack where the attackers can collaborate both by exchanging information about the location of the target and by not marking each other's excessive flows towards the target node. Even under this adverse attack scenario, our algorithm seems to detect between 70% and 100% of all attack traffic in high-volume attacks when

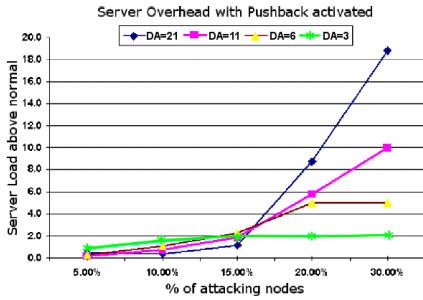


Fig. 8. Increase in server load when under attack and with pushback protocol activated. A load of value 2 means that the server is serving twice the amount of normal requests. The performance of the pushback protocol remains acceptable even when 15% of the overlay nodes are attacking the target.

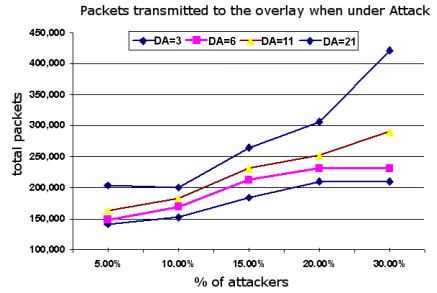


Fig. 9. Total number of packets transmitted to the overlay network, this includes packets that traversed even one hop and then where dropped. Again, the performance of the system is not affected significantly by the attack even when 15% of the overlay nodes are subverted.

even up to 15% of all overlay nodes are participating in the attack, as shown in Figure 7. When 25% of the overlay nodes are participating in the attack, our mechanism correctly identifies between 40% and 50% of all attack traffic. As the percentage of attacking nodes in the population increases, the effectiveness of our mechanism decreases. This, however, should be expected: the notion of “normalcy” in such a system is shifting towards higher-volume traffic flows. Furthermore, as we have anticipated, given more knowledge, attackers can avoid detection especially when the attack is highly distributed ($> 25\%$ of the total nodes are compromised) or the attack is of low intensity ($D_A < 6$).

5.4 Pushback Protocol Performance

In this section we quantify the performance of the simple pushback protocol we devised: we run our experiments enabling nodes to rate limit the aggregate flow that contains the malicious flow using probabilistic rate limiting. The dropping probability is determined based on the deviation of the aggregate flow from the average making the flow conform to within δ from the measured average. Moreover, the node detecting the attack notified its corresponding incoming neighbor of the problem denoting also the sets of key(s) that were affected by this excessive aggregate flow. If the node was unresponsive to the rate limiting request for more than 100 packets (i.e. the rate of the detected aggregate flow remained above the average) this node was immediately marked as malicious and was removed from the overlay by being replaced with a non-attacking node to avoid reconstruction of the overlay routing table. The selection of the 100 packet threshold is a parameter of our system and allows a little more time to the neighbor to adjust to the rate limiting request. It cannot be exploited by attackers since its too low to make a significant impact to the target server.

The most important metric for the performance of our system is the additional load imposed on the attacked node. Figure 8 shows that the load imposed on that node during an attack for various attack intensities remains low, namely below 2 and when the

percentage of the attackers is below 15%. If the fraction of the attacking nodes is increased beyond 15%, our system can only provide partial protection and the load of the server increases dramatically.

Figure 9 shows the total number of packets transmitted to the overlay network, this includes packets that traversed even one hop and then where dropped. Again, the performance of the system is not affected significantly by the attack even when 15% of the overlay nodes are subverted.

6 Conclusion and Future Work

We have examined the problem of distributed denial of service (DDoS) attacks in peer to peer (P2P) systems that exhibit certain properties due to their structured way they forward packets. This include a broad range of DHT systems including locality aware PASTRY and KADEMLIA [15]. However, our method does not apply to CAN since the traffic arriving from various neighbors cannot be well defined. To our knowledge, this is the first work that examines the problem of insider DDoS attacks in such systems. We presented a distributed and scalable mechanism for identifying anomalous traffic flows. Our main intuition was to exploit certain invariant characteristics of a well behaved DHT-based P2P system, in particular, the tendency of aggregate traffic flows through a node to exhibit roughly equal loads. Attackers that introduce (or drop) excessive traffic are identified by measuring the deviation of their flows from the behavior of other flows, as seen by any node in the system. Our simulations show that our mechanism can be extremely effective, detecting attack traffic with 100% accuracy. Using different attack scenarios applied in real Web traces from IRCache [1], we show that our detection can protect against both uncoordinated and coordinated attacks. For uncoordinated DDoS attacks, our algorithm detects most of the attack traffic even when up to 25% overlay nodes are participating in the attack. In the case of coordinated DDoS attacks, the worst type of attack possible, our mechanism works even when up to 15% of the nodes have been subverted.

Our detection and reaction mechanism is fully distributed: it does not depend on any particular node for its operation. When an attack is detected, information about it is “pushed back” to the incoming neighbors that are better positioned to differentiate misbehaving flows. Such nodes are alerted about the attack and recursively apply our approach. We believe that our investigation opens up an important new area in DHT P2P networks. Future directions include the application of our method in localized PASTRY, KADEMLIA and other systems that have structure and evaluation in a real-world (not simulated) environment.

References

1. Ircache web trace repository, <http://www.ircache.net>
2. Andersen, D., Balakrishnan, H., Kaashoek, M., Morris, R.: Resilient Overlay Networks. In: SOSP (October 2001)
3. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S.: Construction of an efficient overlay multicast infrastructure for real-time applications. In: INFOCOM (April 2003)

4. Castro, M., Drushel, P., Ganesh, A., Rowstron, A., Wallach, D.: Secure routing for structured peer-to-peer overlay networks. In: OSDI (2002)
5. Chen-Nee Chuah, R.H.K., Subramanian, L.: DCAP: Detecting Misbehaving Flows via Collaborative Aggregate Policing, vol. 33(5) (October 2003)
6. Dabek, F., Kaashoek, F., Morris, R., Karger, D., Stoica, I.: Wide-area cooperative storage with cfs. In: SOSP (October 2001)
7. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: SOSP (October 2001)
8. Zhao, B.Y., et al.: Tapestry: A Global-scale Overlay for Rapid Service Deployment. IEEE Journal on Selected Areas in Communications, Special Issue on Service Overlay Networks (January 2004)
9. Kubiawicz, J., et al.: OceanStore: An Architecture for Global-scale Persistent Storage. In: ASPLOS (November 2000)
10. Ioannidis, J., Bellovin, S.M.: Implementing Pushback: Router-Based Defense Against DDoS Attacks. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (February 2002)
11. Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, M.F., O’Toole Jr., J.W.: Overcast: Reliable multicasting with an overlay network. In: OSDI, October 2000, pp. 197–212 (2000)
12. Keromytis, A.D., Misra, V., Rubenstein, D.: SOS: Secure Overlay Services. In: SIGCOMM, pp. 61–72 (August 2002)
13. Li, Z., Mohapatra, P.: QRON: QoS-aware routing in overlay networks. IEEE Journal on Selected Areas in Communications, Special Issue on Service Overlay Networks (January 2004)
14. Matrawy, A., Oorschot, P.C.: v., Somayaji, A.: Mitigating Network Denial-of-Service Through Diversity-Based Traffic Management. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 104–121. Springer, Heidelberg (2005)
15. Maymounkov, P., Mazieres, D.: Kademia: A peer-to-peer information system based on the xor metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, Springer, Heidelberg (2002)
16. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: SIGCOMM (August 2001)
17. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
18. Rowstron, A.I.T., Druschel, P.: Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In: SOSP, pp. 188–201 (October 2001)
19. Stoica, I., Adkins, D., Zhuang, S., Shenker, S., Surana, S.: Internet indirection infrastructure. IEEE/ACM Trans. Netw. 12(2), 205–218 (2004)
20. Tran, D.A., Hua, K., Do, T.: A peer-to-peer architecture for media streaming. IEEE Journal on Selected Areas in Communications, Special Issue on Service Overlay Networks (January 2004)

A Appendix

A.1 Chord Invariants and Tests

Proposition 3 *Let $i_1 > i_2$. If the set of flows (S_{i_2}, K) that pass through c is non-empty, then:*

$$\alpha(c, K, S_{i_1}, S_{i_2}) = 2^{i_1 - i_2}.$$

Proof. Assume the set of flows (S_{i_2}, K) that pass through c is non-empty. For each key $k \in K$, let us count the number of sources whose transmissions would pass through node c , entering node c via the i_2 -th finger of node $c - 2^{i_2}$ en-route to key k . Recall that the Chord forwarding protocol requires that a transmission be forwarded on the finger that takes the transmission as far as possible in the clockwise direction without passing the destination. Since the i_2 -th finger travels a distance of 2^{i_2} around the ring, all transmissions prior to reaching node c must traverse a distance greater than 2^{i_2} , and the sequence of fingers taken after reaching c to then reach k is unique.

Consider a sequence of ℓ bits, $b_1 b_2 \dots b_\ell$, where $\ell = \log(N) - i_2$, and consider the node s that is at distance $\sum_{j=1}^{\ell} b_j 2^{j+i_2}$ from c in the counter-clockwise direction. Then, for node s to reach k , it will first traverse a series of nodes where it exits through the $j + i_2$ -th outgoing finger of some node along the path when and only when $b_j = 1$. After taking this series of fingers, the transmission will end up at node c by taking the i_2 -th finger of node $c - 2^{i_2}$ en-route to its final destination of k .

Each of the 2^ℓ possible bit sequences produces a unique node s . Hence, there are 2^ℓ such nodes whose transmissions to k first traverse a path that proceeds through $c - 2^{i_2}$ and takes its i_2 -th finger to reach c en-route to k . Furthermore, since the only way a transmission can originate at a source s and reach k by taking the i_2 -th finger of $c - 2^{i_2}$ is to previously take a strictly decreasing sequence of fingers, this set of 2^ℓ nodes is unique.

It follows that there are $2^{\log(N)-i_2}$ sources that can reach key k by transiting through the i_2 -th finger of $c - 2^{i_2}$, and there are $2^{\log(N)-i_1}$ sources that can reach k by transiting through the i_1 -th finger of $c - 2^{i_1}$, so there are $2^{i_2-i_1}$ times as many sources entering the i_1 -th finger of c to reach k as there are entering c through its i_2 -th incoming finger to reach k .

Since we assume that all sources have (approximately) the same interest in key k , and since this ratio remains fixed irrespective of the final distance of key k from c , the Lemma holds.

A.2 Evaluation Results

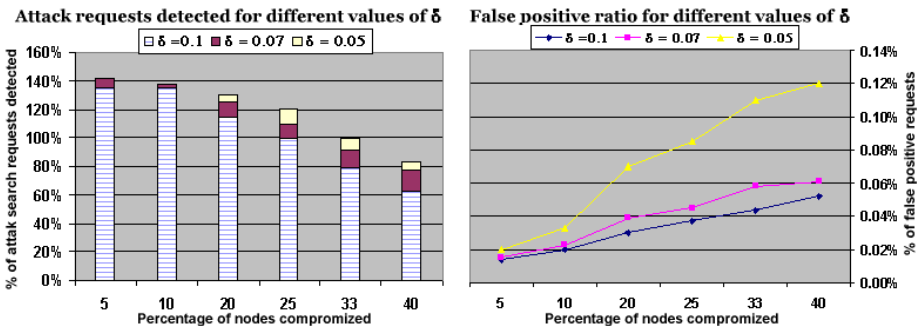


Fig. 10. The left graph shows the increase in the detection rate as we decrease our tolerance threshold from $\delta = 0.1$ to $\delta = 0.05$. The right graph shows the impact of this increase in the false positive percentage: lowering δ below 0.07 offers little benefit to the detection rate while doubling false positives.

PPAA: Peer-to-Peer Anonymous Authentication^{*}

Patrick P. Tsang and Sean W. Smith

Department of Computer Science
Dartmouth College
Hanover, NH 03755, USA
{patrick,sws}@cs.dartmouth.edu

Abstract. In the pursuit of authentication schemes that balance user privacy and accountability, numerous anonymous credential systems have been constructed. However, existing systems assume a client-server architecture in which only the clients, but not the servers, care about their privacy. In peer-to-peer (P2P) systems where both clients and servers are peer users with privacy concerns, no existing system correctly strikes that balance between privacy and accountability.

In this paper, we provide this missing piece: a credential system in which peers are *pseudonymous* to one another (that is, two who interact more than once can recognize each other via pseudonyms) but are otherwise anonymous and unlinkable across different peers. Such a credential system finds applications in, e.g., Vehicular Ad-hoc Networks (VANets) and P2P networks.

We formalize the security requirements of our proposed credential system, provide a construction for it, and prove the security of our construction. Our solution is efficient: its complexities are independent of the number of users in the system.

Keywords: privacy, anonymous authentication, credentials, secret handshakes, VANets, reputation systems.

1 Introduction

We live in an era where human activities happen electronically more than ever. People rely heavily on computer infrastructures, such as Web applications and peer-to-peer (P2P) networks, to share information, express opinions and trade goods. It is therefore paramount to protect the privacy of the users in these infrastructures by providing them with the option of acting anonymously, unlinkably and/or unobservably.

^{*} This work was supported in part by the National Science Foundation under grant CNS-0524695. The views and conclusions do not necessarily represent those of the sponsors.

1.1 Balancing User Privacy and Accountability

It is impractical to pursue user privacy without taking accountability into consideration. Without the fear of being identified, held responsible and punished when they abuse the services, clients are likely to misbehave due to selfishness or malice, thereby disrupting system operations and harming everyone else. Accountability has traditionally been achieved through authentication mechanisms (often followed by access control and/or auditing), which verify the identity of a client who requests a service. In the classic examples of passwords, Kerberos and standard Public Key Infrastructures (PKIs), clients have to give up their privacy to be authenticated.

Anonymous Credential Systems. In the pursuit of authentication schemes that balance privacy and accountability, numerous anonymous credential systems [15, 17], and closely related schemes such as k -times anonymous authentication (k -TAA) [33, 34], offline anonymous electronic cash (e-cash) systems [20, 14] and group signatures [21, 5] have been constructed. An anonymous credential system allows a client to be authenticated by a server as a group member anonymously and unlinkably, and yet the anonymity can be revoked when certain conditions are met. Existing systems differ in their anonymity revocation mechanisms, and hence provide different balancing points between privacy and accountability for different application settings. For example, clients can be identified when they “double-spend” in an e-cash system; their authentications become linkable¹ when they are authenticated more than k times in k -TAA. In group signatures, an authority exists and is capable of arbitrarily revoking anonymity.

1.2 The Challenge: P2P Systems

All anonymous credential systems in existence today assume a client-server architecture in which only the clients, but not the servers, care about their privacy. However, in P2P systems where both clients and servers are peer users with privacy concerns, none of the existing credential systems correctly strike that balance between privacy and accountability.

More specifically, several existing anonymous credential systems provide client accountability by empowering servers to pseudonymize clients who are otherwise anonymous, and servers can thus decide whether and/or how to serve an anonymous client depending the past behavior of the client. In all such systems, however, a client must either (1) present to all servers the very same and hence linkable pseudonym, or (2) learn the identity or at least the pseudonym of a server and then present to that server a pseudonym specific to it. In the former case, client privacy is at risk because colluding servers can link connections from the same client; in the latter, server privacy is at risk because colluding clients can link connections to the same server.

¹ Two authentication runs are linkable (by some entity) *if and only if* it is possible (for that entity) to tell whether or not the two runs are executed by the same client.

We provide below two application scenarios to motivate the user's need for privacy not just as a client, but also as a server, in P2P systems. The opposing requirements of server privacy, client privacy, server accountability and client accountability in these scenarios illustrate the non-triviality of the challenge we overcome in this paper.

Vehicular Ad-Hoc Networks (VANets). To contribute to safer and more efficient roads, vehicles in VANets constantly exchange information such as road and weather conditions among each other and with roadside base stations. Research has shown that the provision of the necessary security and privacy in VANets is critical to the users who rely on these networks [30, 12].

To protect the location privacy of the drivers when information is exchanged on the road between two vehicles, both vehicles should remain anonymous among all the vehicles in communication range. Furthermore, no one should be able to link reports by the same vehicle to different other vehicles or roadside base stations. This helps prevent a vehicle from being not only pseudonymized and thus tracked, but also deanonymized through drawing inferences from multiple reports made by the vehicle [28].

From the accountability perspective, to distinguish legitimate data from rogue data, vehicles must be authenticated when reporting sensor readings. Moreover, so that repetitive reporting of the same information can be detected, vehicles should be pseudonymous to one another (that is, vehicle X can recognize some vehicle Y reporting again, without knowing anything else about Y). For instance, in VANets in which vehicles decide when to accelerate and break based on reports collected from the network, the failure to achieve these security goals can allow an attacker to paralyze traffic and/or induce accidents.

Reputation Systems for P2P Networks. The existence of selfish users in P2P networks such as those for file sharing severely degrades system performance. Adversaries can reduce the availability of specific items in P2P networks by "poisoning" [22] them, i.e., injecting lots of decoys into the network. Reputation systems provide a game-theoretic solution to these problems by introducing incentives for users to behave well. Unfortunately, reputation systems lacking privacy can also introduce disincentives to good behavior: if a reputation system reveals the pseudonym or even the identity of the serving peers, peers might refuse to serve others so as to stay anonymous.

A privacy-preserving reputation system for P2P networks where there is no (trusted) central server should have the following properties: users are pseudonymous to one another, so that a user Carol can decide whether to serve (or be served by) another user Dave based on her past experience with Dave, without knowing his actual identity. However, assuming the registration procedures make sure that users in the system can have at most one single membership, Dave shouldn't be able to start fresh after having established a bad reputation with respect to Carol, nor can he impersonate Carol for her high reputation, potentially even spoiling her reputation through misbehavior. Finally, connections

between a peer Carol and different other peers should be unlinkable, as otherwise it might be possible for someone to trace Carol by studying those connections.

1.3 Our Contributions

In this paper, we overcome the challenge posed above by proposing the concept—and giving a construction and implementation—of *Peer-to-Peer Anonymous Authentication*, or PPAA for short, a credential system in which peers are pseudonymous to individual peers but unlinkable across different peers. More specifically, we make the following contributions:

- We rigorously define the operations of PPAA and its security and privacy requirements, during which we introduce the notion of the *Linkability Context* of an authentication scheme as a tool for a more precise reasoning about the linkability property of an authentication scheme. We also formalize the threat model in which those security requirements must be satisfied.
- We provide the first construction for PPAA. Our construction is both secure and efficient. In particular, its complexities are independent of the number of users in the system. In the extended version of this paper [36], we also report empirical performance figures of a software implementation of our construction.

Paper Organization. We review the related works in Section 2 and give an overview of our solution in Section 3. Section 4 covers the preliminary materials. In Section 5, we define the security model. We present our solution and analyze its security and efficiency in Section 6. We provide some discussions in Section 7 and conclude the paper in Section 8.

2 Related Works

We review the literature for related works, and argue why they fail to solve the problem posed in this paper. We make occasional but otherwise minimal use of mathematical notation without definition for the sake of conciseness.

***k*-Times Anonymous Authentication.** *k*-TAA [33, 29, 34, 13] and related schemes such as event-oriented linkable group/ring signatures [37, 6] are close candidates in overcoming the posed challenge. In essence, when a client Alice in these schemes is being authenticated by a server Bob, she provides Bob with a tag and convinces him that the tag is correctly formed. Bob can test if two authentications are linked to the same client by examining the associated tags.

These schemes do not solve the posed problem since authentication runs by the same user to different servers are linkable. This is because a user always uses the same tag when being authenticated by any server. More specifically, the tag of client i with secret x_i has the form of $t_i = g^{x_i}$ for some global parameter g .

We point out that, while they do not address server privacy, various *k*-TAA schemes and anonymous credential systems do provide several major ingredients

for the solution we propose in this paper. For example, our proof system for group membership uses ideas from Camenisch and Lysyanskaya [17] and Boneh et al. [10]. Also, the concept of event identifiers in this paper stems from several other existing schemes [15, 33, 37].

Secret Handshakes. Secret handshake schemes (SHSs) [7, 19, 40, 4] allow any two members of the same group to authenticate each other as a group member and share a session key without revealing their group affiliations to outsiders.

In the scheme due to Xu and Yung [40], secret handshakes are anonymous and unlinkable, but members are limited to shaking hands no more than some predefined number of times. The state-of-the-art construction [4] provides anonymity and unlinkability without such a limitation. Recently, Tsudik and Xu [38] extended secret handshakes into a multi-party and privacy-conserving setting: two or more group members can anonymously and unlinkably authenticate each other such that one’s group affiliation is not revealed unless every other party’s membership is ensured.

All anonymous secret handshakes proposed so far [40, 4, 38] fail to solve the posed problem. As handshakes are unlinkable, a client Alice has no way to tell if the one she is shaking hands with is the same as the one behind some earlier handshakes. As a remedy, Alice may ask the person behind the handshake to reveal a secret, e.g., a random nonce, that she leaked in their last handshake. Unfortunately, this is problematic because the person does not know which secret to reveal as Alice is anonymous. Also, one could pretend to be new by “forgetting” the secrets.

3 Our Approach

In this section, we provide an overview of our approach to solve the posed challenge.

3.1 Putting Authentication Schemes into “Linkability Context”

We first introduce the notion of the *linkability context* in authentication.

Definition 1 (Linkability Context). The *Linkability Context*, or *LC* for short, of an authentication scheme is a collection of attributes that determines the linkability of authentication runs in the scheme. In particular, two authentication runs are *linkable if and only if* the two runs are executed when the attributes in the linkability context are all in the same condition. \square

In k -TAA, for instance, authentication runs by the same client at the same “time” are linkable, while runs by the same client at different times, as well as those by different clients at the same time, are not linkable. The linkability context of k -TAA is thus $LC = \{\text{client-ID}, \text{time}\}$, i.e., the collection of client identity and time.

Understanding the precise linkability context of an authentication scheme helps reason about the privacy guarantees and hence implications of the scheme. At one end of the spectrum of client privacy, in conventional authentication

schemes such as those using digital signatures, any two authentication runs are linkable. The linkability context of these schemes therefore consists of nothing, i.e., $LC = \emptyset$. At the other end of the spectrum, there are schemes such as ring authentication [31, 23] in which no two authentication runs are linkable. In this case, the linkability context is the authentication run instance, i.e., $LC = \{\text{authen-run-ID}\}$.

Linkability Context in PPAA. A correct choice of its linkability context is the first step towards a secure PPAA construction. In our design, the linkability context in PPAA is the collection of the *unordered* pair of client and server identity, and the event for which the PPAA authentication is executed, i.e.,

$$LC = \{\{\text{client-ID}, \text{server-ID}\}, \text{event-ID}\}.$$

In other words, we would like to design PPAA in such a way that authentication runs are linkable *if and only if* they are executed between the same pair of peers for the same event. In the example of VANets, if one sets the event to be “*speed on Highway I-89 on June 3rd, 2008*,” then only those PPAA-authenticated speed report made by the same vehicle to the same road-side base station on Highway I-89 on June 3rd, 2008 are linkable.

3.2 Key Ideas in Our PPAA Design

An Observation. It should have become clear now that event-oriented linkable group/ring signatures and k -TAA fail as a secure PPAA construction because server identity is not in their linkability context. It would seem that one could bring server identity into the linkability context in an event-oriented linkable group/ring signature (resp. k -TAA) by mapping an event in (resp. one “time”) into the identity of a server. Consequently, LC becomes $\{\text{client-ID}, \text{server-ID}\}$ and authentication runs by the same user to different servers become unlinkable. More specifically, the tag of client i with secret x_i with respect to server j has the form of $t_{i,j} = g_j^{x_i}$, where g_j is a server-specific parameter. Tags of the same client with respect to different servers are now unlinkable thanks to the underlying intractability assumption (the Decisional Diffie-Hellman assumption).

Unfortunately, to produce a tag and prove its correctness during an authentication run in the above modified scheme, a client must now ask the server for its g_j , which can be considered its pseudonym. Even if there existed a way in which a client could compute a tag for the server without knowing the pseudonym of the server, two colluding users can easily determine if they are being authenticated by the same server. In other words, using the tag design in event-oriented linkable group/ring signatures and k -TAA, it is impossible to devise a secure authentication scheme with $LC = \{\text{client-ID}, \text{server-ID}\}$.

The Need of a Novel Tag Construct. As a result, constructing a secure PPAA requires a new tag design that possesses novel features:

- Tags must be dependent on the identity of the client, the server, and the event.

- Tags are linked *if and only if* they are produced by the same (unordered) pair of peers, and during the same event.
- Peers must be able to produce tags and prove their correctness in zero-knowledge through interacting with the other peers and without knowing the identity of the other peers.

In Section 6, we present such a tag design and how we use it to construct a secure PPAA.

4 Preliminaries

We provide the technical background necessary for understanding the rest of this paper.

Notations. A function $f(\lambda)$ is negligible if for all polynomial $p(\lambda)$, $f(\lambda) < 1/p(\lambda)$ holds for all sufficiently large λ . A function is non-negligible if it is not negligible. The probability $\Pr[E]$ of an event E is overwhelming (in some parameter λ) if $1 - \Pr[E]$ is negligible (in λ).

Let λ be a sufficiently large security parameter. Let \mathbb{G}_1 and \mathbb{G}_2 be cyclic groups of prime order p with $|p| = \lambda$ such that group operation is efficiently computable. Let g_0 and h_0 be generators of \mathbb{G}_1 and \mathbb{G}_2 respectively such that there is an efficiently computable isomorphism ψ from \mathbb{G}_2 to \mathbb{G}_1 with $\psi(h_0) = g_0$.

We say that $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair if there exists an efficiently computable map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where \mathbb{G}_T is also a cyclic group of prime order p , such that: $\hat{e}(A^x, B^y) = \hat{e}(A, B)^{xy}$ for all $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_p$, and $\hat{e}(g_0, h_0) \neq 1$.

Complexity Assumptions. The security of our solution to be presented later in this paper relies on the validity of the DDH assumption in \mathbb{G}_1 and the q -SDH assumption on bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, which we define as the following.

- *The Decisional Diffie-Hellman (DDH) problem* in \mathbb{G}_1 : On input of a quadruple $(g_0, g_0^a, g_0^b, g_0^c) \in \mathbb{G}_1^4$, where $a, b \in_R \mathbb{Z}_p$, and $c = ab$ or $c \in_R \mathbb{Z}_q$ equally likely, output 1 if $c = ab$ and 0 otherwise. We say that *the DDH assumption* in \mathbb{G}_1 holds if no probabilistic polynomial time (PPT) algorithm has non-negligible advantage over random guessing in solving the DDH problem in \mathbb{G}_1 .
- *The q -Strong Diffie-Hellman (q -SDH) problem* in $(\mathbb{G}_1, \mathbb{G}_2)$: On input of a $(q + 2)$ -tuple $(g_0, h_0, h_0^x, h_0^{x^2}, \dots, h_0^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, where $x \in_R \mathbb{Z}_p$, output a pair $(A, c) \in \mathbb{G}_1 \times \mathbb{Z}_p$ such that $A^{(x+c)} = g_0$. We say that *the q -SDH assumption* in $(\mathbb{G}_1, \mathbb{G}_2)$ holds if no PPT algorithm has non-negligible advantage in solving the q -SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

The q -SDH assumption was introduced and proven to hold in generic groups [32] by Boneh and Boyen [9]. The DDH assumption in \mathbb{G}_1 is also known as the *eXternal Diffie-Hellman (XDH)* assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ [14, 10]. The validity of the XDH assumption implies that ψ is computationally one-way. The assumption

is known is false on supersingular curves [25], but is conjectured to hold for the Weil or Tate pairing on MNT curves with embedded degree greater than 1 and \mathbb{G}_1 defined over the ground field [10].

Proofs of Knowledge. In a *Zero-Knowledge Proof-of-Knowledge (ZKPoK)* protocol [26], a prover convinces a verifier that some statement is true without the verifier learning anything except the validity of the statement. Σ -protocols are a special type of three-move ZKPoK protocols. They can be converted into non-interactive *Signature Proof of Knowledge (SPK)* schemes that are secure in the *Random Oracle (RO)* Model [8] (in the sense of Indistinguishability against chosen-message attacks, or IND-CMA [27]).

In many anonymous credential systems, a client uses an SPK scheme to prove in zero-knowledge to a server her possession of a credential issued by the Group Manager when being authenticated by a server. The SPK schemes differ in these systems, which accounts for the differences in privacy and accountability guarantees and complexity assumptions. The SPK schemes we will use in our solution are based on the ZKPoK protocol due to Boneh and Boyen [10].

We follow the notation introduced by Camenisch and Stadler [18] for the various ZKPoK protocols. For example, $PK \{(x) : y = g^x\}$ denotes a ZKPoK protocol that proves the knowledge of an integer x such that $y = g^x$ holds, where y and g are elements of some group $G = \langle g \rangle$. Using this notation, a ZKPoK protocol can be described by just pointing out its aim while hiding all the details. Moreover, we denote by $SPK \{(x) : y = g^x\}(M)$ the SPK scheme converted from the above ZKPoK protocol.

5 Model

This section formalizes PPAA. The entities involved in PPAA are the Group Manager (GM) and a set of peer users, or simply peers. The GM is responsible for registering peers. A peer can be a client, a server, or both. Clients are interested in accessing services provided by servers and servers are willing to serve the clients, as long as their privacy and accountability requirements are satisfied.

5.1 System Operations

Operations that take place in PPAA include the GM setting up the system (**Setup**) and registering peers into the system (**Registration**), and peers authenticating one another (**Authentication**) and testing if two authentication runs are linked (**Linking**). We highlight that only **Setup** and **Registration** involve a centralized authority, namely the GM; **Authentication** requires no centralized authority, which is a crucial property necessary for PPAA to be applicable to P2P systems with scalability.

The syntax for these operations are given as follows.

- **Setup** is a *Probabilistic Poly-Time (PPT)* algorithm invoked by the GM. On input a sufficiently large security parameter λ , the algorithm outputs GM's

secret key \mathbf{gsk} and the group public key \mathbf{gpk} . The GM stores \mathbf{gsk} privately and publishes \mathbf{gpk} to the public. \mathbf{gpk} is an implicit input to all the algorithms below.

- **Registration** is a two-party multi-round protocol between the Register^P PPT algorithm invoked by a peer and the Register^{GM} PPT algorithm invoked by the GM. The additional input to Register^{GM} is the GM's secret key \mathbf{gsk} . Upon successful termination of a protocol run, Register^P outputs a credential, which the peer stores privately, and by doing so becomes a registered peer in the system.
- **Authentication** is a two-party multi-round protocol between the Authenticate^I PPT algorithm invoked by a registered peer Alice (as the *Initiator*, i.e. the one who initiates the protocol) and the Authenticate^R PPT algorithm invoked by another registered peer Bob (as the *Responder*). The common input to both parties is an event identifier \mathbf{eid} upon which they have already agreed.² The additional inputs to Authenticate^I and Authenticate^R are Alice's credential and Bob's credential, respectively.

A protocol run terminates successfully *if and only if* both algorithms output a tag, in which case we say that the authentication is *successful* and that Alice and Bob are mutually authenticated with one another, during an event with identifier \mathbf{eid} . When we say that a peer Carol is involved in an authentication without specifying her role, then Carol can be either the initiator or the responder in that authentication.

- **Linking** is a (possibly probabilistic) poly-time algorithm any peer can invoke. On input two tags \mathbf{tag}_1 and \mathbf{tag}_2 , the algorithm outputs a boolean value of either `linked` or `not-linked`.

In the former (resp. the latter) case, the two tags, and also the two successful authentication runs from which the tags are resulted, are said to be *linked* (resp. *not linked*).

Semantically, a peer Carol uses this algorithm to pseudonymize other peers with which she has mutually authenticated: for any two successful authentication runs during the same event, she thinks she is mutually authenticating with the same peer *if and only if* the two authentication runs are linked.

Any construction of PPAA must be correct:

Definition 2 (Correctness). An PPAA construction is correct if it has *authentication correctness* and *linking correctness*:

- *Authentication Correctness.* If all entities in PPAA are honest (i.e. they all follow the system's specification), then, with overwhelming probability, any authentication between any two registered peers is successful.
- *Linking Correctness.* If all entities in PPAA are honest, then, with overwhelming probability, in any two successful authentication involving any registered peer Carol, the two tags output by Carol are linked *if and only if*, in those two authentications, both the event identifiers and the other peers involved are identical. \square

² In the VANet example given in Section 3.2, the \mathbf{eid} can be 20080603||I-89||speed.

5.2 Security Requirements

Roughly speaking, a PPAA construction is secure if it satisfies the following security requirements. (A formal definition can be found in the extended version of this paper [36].)

Mis-authentication Resistance. Mis-authentication occurs when two peers successfully complete mutual authentication, but only one of them is an honest and registered peer. A secure PPAA construction must be resistant to mis-authentication.

For example, this property prevents vehicles in VANets from believing (malicious) data from rogue sensors.

Peer Accountability. To subvert peer accountability, a coalition of $n \geq 1$ registered but malicious peer(s) attempts to run more than n successful mutual authentication involving the same honest peer Carol during the same event such that the tags Carol outputs in those authentication are all pairwise unlinked. A secure PPAA construction requires that no adversary can succeed in such an attempt.

In the example of P2P networks, this prevents a peer from starting fresh after having established a bad reputation with respect to another peer.³

Peer Privacy. To subvert the privacy of an honest peer Carol involved in an authentication potentially executed with a malicious peer, the adversary, potentially with the GM's help, attempts to:

- deanonymize Carol in individual protocol runs, and/or
- pseudonymize Carol in protocol runs with different peers and/or during different events.

A secure PPAA construction requires that no adversary can succeed in any of the above attempts.

As an example, this ensures that communications of a vehicle in VANets with different other vehicles or roadside base stations cannot be linked.

Framing Resistance. An honest peer Carol is *framed* when another honest peer Dave thinks that he is mutually authenticating with the same peer in two successful authentication runs, even though Carol is involved in exactly one of them. A secure PPAA construction requires that no adversary, even with the help of the GM, can frame an honest peer.

In the example of P2P reputation systems, this makes sure that peers can't impersonate other peers with high reputation.

6 Our Solution

We begin this section with a presentation of our first attempt to construct PPAA, which, although insecure by itself, illustrates our tag design as the core component of our full and secure PPAA construction. Then we proceed to present our actual PPAA construction. In the extended version of this paper [36], we discuss our implementation of the construction and its empirical performance evaluation.

³ This assumes that a peer can't register more than once. We will discuss this issue further in Section 7.

6.1 Our First Attempt

We call our first attempt Basic-PPAA.

Parameters. Let \mathbb{G}_1 be a group as described in Section 4 in which the DDH assumption holds. Let $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ be a secure cryptographic hash function. Event identifiers are strings of any length.

Credentials. Each user is given by the GM one credential \mathbf{cred} in the form $\mathbf{cred} = (A, x, y) \in \mathbb{G}_1 \times \mathbb{Z}_p^2$, where $x, y \in_R \mathbb{Z}_p$ and A is distinct in all credentials.

Tags. In Basic-PPAA, a tag is the output of a function f that takes as inputs the credential of an initiating peer $\mathbf{cred}_1 = (A_1, x_1, y_1)$, the credential of a responding peer $\mathbf{cred}_2 = (A_2, x_2, y_2)$ and an event identifier \mathbf{eid} . The function is defined as follows:

$$f : (\mathbf{cred}_1, \mathbf{cred}_2, \mathbf{eid}) \mapsto \mathbf{tag} \doteq \{\tau_1, \tau_2\}, \text{ where } \begin{cases} \tau_1 = A_1^{x_2} H(\mathbf{eid})^{y_1}, \\ \tau_2 = A_2^{x_1} H(\mathbf{eid})^{y_2}. \end{cases}$$

Thus, a tag is a set of two \mathbb{G}_1 elements.

The Skeleton Protocol. The following steps describe a protocol run between an initiating peer Alice with credential $\mathbf{cred}_1 = (A_1, x_1, y_1)$ and a responding peer Bob with credential $\mathbf{cred}_2 = (A_2, x_2, y_2)$ during an event with identifier \mathbf{eid} . When the protocol terminates, Alice and Bob output a tag.

1. Alice \rightarrow Bob: $\langle U_1, V_1 \rangle = \langle A_1^{r_1}, H(\mathbf{eid})^{r_1} \rangle$, where $r_1 \in_R \mathbb{Z}_p$.
2. Bob \rightarrow Alice: $\langle U_2, V_2, W_2 \rangle = \langle A_2^{r_2}, H(\mathbf{eid})^{r_2}, U_1^{x_2} V_1^{y_2} \rangle$, where $r_2 \in_R \mathbb{Z}_p$.
3. Alice \rightarrow Bob: $\langle W_1, \tau_1 \rangle = \langle U_2^{x_1} V_2^{y_1}, W_2^{1/r_1} \rangle$.
4. Bob \rightarrow Alice: $\langle \tau_2 \rangle = \langle W_1^{1/r_2} \rangle$.
5. Alice and Bob both output $\mathbf{tag} = \{\tau_1, \tau_2\} = f(\mathbf{cred}_1, \mathbf{cred}_2, \mathbf{eid})$ and terminate.

Properties. The tags and the skeleton protocol given above have the following desirable properties:

1. Two tags $\mathbf{tag} = f(\mathbf{cred}_1, \mathbf{cred}_2, \mathbf{eid})$ and $\mathbf{tag}' = f(\mathbf{cred}'_1, \mathbf{cred}'_2, \mathbf{eid}')$ are the same *if and only if* $\{\mathbf{cred}_1, \mathbf{cred}_2\} = \{\mathbf{cred}'_1, \mathbf{cred}'_2\}$ and $\mathbf{eid} = \mathbf{eid}'$, with overwhelming probability.
2. The protocol view of Alice $\langle \mathbf{cred}_1, \mathbf{eid}, r_1, U_2, V_2, W_2, \tau_2 \rangle$ can be simulated (computationally indistinguishably) by Alice if she is given \mathbf{tag} . In other words, Alice learns no knowledge other than \mathbf{tag} from running the skeleton protocol. Similarly, Bob learns no knowledge other than \mathbf{tag} from running the skeleton protocol.
3. The tag produced by a peer Alice for another peer Bob during an event is indistinguishable from the tag produced by any peer for Bob during a different event; it is also indistinguishable from the tag produced by Alice for a different peer during the same event.

The validity of these properties are straightforward provided that the DDH assumption in \mathbb{G}_1 holds. We thus omit the proof.

Remark. If not all entities are honest, Basic-PPAA results in an insecure PPAA construction. For instance, users can be authenticated without asking the GM for a credential, dishonest users may use an arbitrary credential instead of the one given by the GM to get away from being linked and a malicious GM can frame clients.

6.2 Our PPAA Construction

We now enumerate our PPAA construction. It can be thought of as the result of securing Basic-PPAA by adding to it all necessary mechanisms to force the entities to behave honestly, such as by accompanying each step in the skeleton protocol with a SPK scheme that proves the correctness of the step.

Parameters. In addition to those in Basic-PPAA, our PPAA construction has the following parameters. Let \mathbb{G}_2 be a group as described in Section 4 such that $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair in which the q -SDH assumption holds. Let ℓ be a sufficiently large security parameter of size polynomial in λ . Let $g_1, \dots, g_5 \in \mathbb{G}_1$ be generators of \mathbb{G}_1 such that the relative discrete logarithms among g_1, \dots, g_5 and g_0 (from Section 4) are unknown. Let $\hat{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a secure cryptographic hash function. \hat{H} is utilized by the various SPKs in the construction.

Setup. The GM randomly chooses $\gamma \in_R \mathbb{Z}_p$ and computes $w = h_0^\gamma \in \mathbb{G}_2$. The group secret key is $\mathbf{gsk} = (\gamma)$ and the group public key is $\mathbf{gpk} = (w)$.

Registration. At the successful termination of a run of this protocol between a user Alice and the GM, Alice obtains a credential \mathbf{cred} in the form of $\mathbf{cred} = (A, e, x, y, z) \in \mathbb{G}_1 \times \mathbb{Z}_p^4$ such that $A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z$. The private input to the GM is his group secret key \mathbf{gsk} . The protocol proceeds as follows.

1. The GM sends $\langle N_0 \rangle$ to Alice, where $N_0 \in_R \{0, 1\}^\ell$ is a random challenge.
2. Alice sends $\langle C, \Pi_0 \rangle$ to the GM, where $C = g_1^x g_2^y g_3^{z'} \in \mathbb{G}_1$ is a commitment of $(x, y, z') \in_R \mathbb{Z}_p^3$ and Π_0 is a signature proof of knowledge of

$$SPK \left\{ (x, y, z') : C = g_1^x g_2^y g_3^{z'} \right\} (M)$$

on message $M = N_0 || C$, which proves the correctness of C . We will refer to the above SPK as SPK_0 .

3. The GM terminates with **failure** if the verification of Π_0 returns **invalid**. Otherwise the GM sends $\langle A, e, z'' \rangle$ to Alice, where $e, z'' \in_R \mathbb{Z}_p$ and

$$A = (g_0 C g_3^{z''})^{\frac{1}{e+\gamma}} \in \mathbb{G}_1$$

4. Alice computes $z = z' + z''$. She terminates with **failure** if $\hat{e}(A, wh_0^e) \neq \hat{e}(g_0 g_1^x g_2^y g_3^z, h_0)$. Otherwise she outputs $\mathbf{cred} = (A, e, x, y, z)$ as her credential.

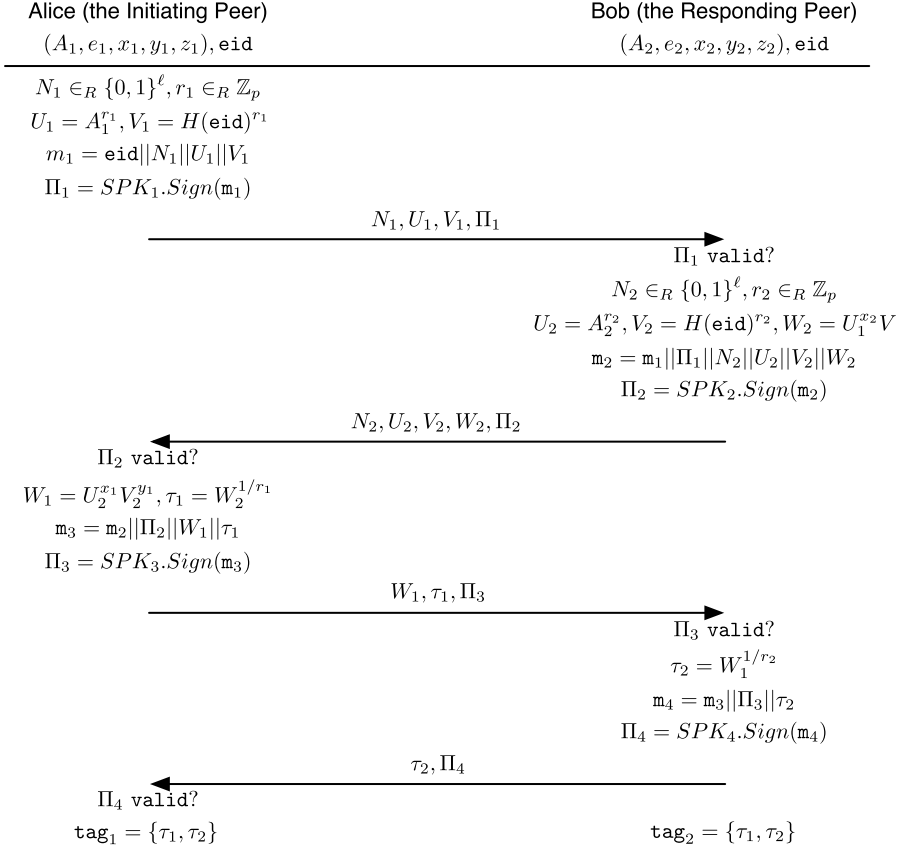


Fig. 1. The Authentication Protocol

We remark that the security of the system requires that no two instances of the Registration protocol may run concurrently. To enforce this rule, the GM registers users one after the other.

Authentication. Alice (as the initiator) and Bob (as the responder) would like to mutually authenticate with each other during an event with identifier $\mathbf{eid} \in \{0, 1\}^*$. The common input to both Alice and Bob is \mathbf{eid} . The private input to Alice and Bob is their own credentials $(A_1, e_1, x_1, y_1, z_1)$ and $(A_2, e_2, x_2, y_2, z_2)$ respectively. The following describes the steps in the 4-round protocol for authentication.

1. Alice sends $\langle N_1, U_1, V_1, \Pi_1 \rangle$ to Bob, where:
 - $N_1 \in_R \{0, 1\}^\ell, r_1 \in_R \mathbb{Z}_p$,
 - $U_1 = A_1^{r_1} \in \mathbb{G}_1, V_1 = H(\mathbf{eid})^{r_1} \in \mathbb{G}_1$, and
 - Π_1 is a signature proof of knowledge of

$$SPK \left\{ (A, e, x, y, z, r) : \begin{array}{l} A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z \\ U_1 = A^r \wedge V_1 = H(\mathbf{eid})^r \end{array} \wedge \right\} (M)$$

on message $M = \mathbf{m}_1 = \text{eid}||N_1||U_1||V_1 \in \{0, 1\}^*$, which Alice can produce using her knowledge of $(A_1, e_1, x_1, y_1, z_1, r_1)$. We will refer to the above *SPK* as *SPK*₁.

2. Bob terminates with **failure** if verification of Π_1 returns **invalid**. Otherwise he sends $\langle N_2, U_2, V_2, W_2, \Pi_2 \rangle$ to Alice, where:

- $N_2 \in_R \{0, 1\}^\ell$, $r_2 \in_R \mathbb{Z}_p$,
- $U_2 = A_2^{r_2} \in \mathbb{G}_1$, $V_2 = H(\text{eid})^{r_2} \in \mathbb{G}_1$, $W_2 = U_1^{x_2} V_1^{y_2}$, and
- Π_2 is a signature proof of knowledge of

$$SPK \left\{ (A, e, x, y, z, r) : \begin{array}{l} A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z \quad \wedge \\ V_2 = H(\text{eid})^r \quad \wedge \quad U_2 = A^r \quad \wedge \\ W_2 = U_1^x V_1^y \end{array} \right\} (M)$$

on message $M = \mathbf{m}_2 = \mathbf{m}_1||\Pi_1||N_2||U_2||V_2||W_2 \in \{0, 1\}^*$, which Bob can produce using his knowledge of $(A_2, e_2, x_2, y_2, z_2, r_2)$. We will refer to the above *SPK* as *SPK*₂.

3. Alice terminates with **failure** if verification of Π_2 returns **invalid**. Otherwise she sends $\langle W_1, \tau_1, \Pi_3 \rangle$ to Bob, where:

- $W_1 = U_2^{x_1} V_2^{y_1} \in \mathbb{G}_1$, $\tau_1 = W_2^{1/r_1} \in \mathbb{G}_1$, and
- Π_3 is a signature proof of knowledge of

$$SPK \left\{ (A, e, x, y, z, r) : \begin{array}{l} A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z \quad \wedge \\ U_1 = A^r \quad \wedge \quad V_1 = H(\text{eid})^r \quad \wedge \\ W_1 = U_2^x V_2^y \quad \wedge \quad W_2 = \tau_1^r \end{array} \right\} (M)$$

on message $M = \mathbf{m}_3 = \mathbf{m}_2||\Pi_2||W_1||\tau_1 \in \{0, 1\}^*$, which Alice can produce using her knowledge of $(A_1, e_1, x_1, y_1, z_1, r_1)$. We will refer to the above *SPK* as *SPK*₃.

4. Bob terminates with **failure** if verification of Π_3 returns **invalid**. Otherwise he sends $\langle \tau_2, \Pi_4 \rangle$ to Alice, where:

- $\tau_2 = W_1^{1/r_2}$, and
- Π_4 is a signature proof of knowledge of

$$SPK \{(r) : W_1 = \tau_2^r \quad \wedge \quad V_2 = H(\text{eid})^r\} (M)$$

on message $M = \mathbf{m}_4 = \mathbf{m}_3||\Pi_3||\tau_2 \in \{0, 1\}^*$, which Bob can produce using his knowledge of (r_2) . We will refer to the above *SPK* as *SPK*₄.

Bob outputs $\text{tag}_2 = \{\tau_1, \tau_2\}$ and terminates.

5. Alice terminates with **failure** if verification of Π_4 returns **invalid**. Otherwise she outputs $\text{tag}_1 = \{\tau_1, \tau_2\}$ and terminates.

Figure 1 is a diagrammatic representation of the protocol.

Linking. On input two tags $\text{tag}_1, \text{tag}_2$, this algorithm returns **linked** if they are equal and **not-linked** otherwise.

6.3 SPK Instantiation

The instantiation of SPK_0 to SPK_4 and their computational costs in terms of the number of pairing computation and multi-exponentiations (multi-EXPs)⁴ can be found in the extended version of this paper [36].

6.4 Analysis

Our PPAA construction has correctness, which is a straightforward consequence of the correctness of the skeleton protocol and the correctness of the various SPK schemes. We omit the proof for conciseness.

Security. The security of our construction hinges on the correctness of the skeleton protocol and the security properties of the various SPK schemes surrounding it. We now state the following theorem. (Its proof is sketched in the extended version of this paper [36].)

Theorem 1 (Security). Our proposed PPAA construction is secure in the random oracle model if the XDH assumption and the q -SDH assumption hold in $(\mathbb{G}_1, \mathbb{G}_2)$. \square

Complexities. Our solution scales extremely well: all operations have constant computational and communication complexities, regardless on the number of peers, events and authentication runs. Registration is a one-time process per user in the system. Linking involves only an equality testing of two sets of two \mathbb{G}_1 elements.

Authentication is the dominating operation, thus we provide a more detailed analysis on its costs. Alice, the initiating peer, needs to do an SPK_1 and an SPK_3 signing, and an SPK_2 and an SPK_4 verification. The number of \mathbb{G}_1 multi-EXPs, \mathbb{G}_T multi-EXPs and pairings are 24, 10 and 4 respectively. Some of these operations can be precomputed before the start of an authentication; with precomputation, those numbers become 10, 4 and 2 respectively. Bob, the responding peer, needs to an SPK_2 and an SPK_4 signing, and an SPK_1 and an SPK_3 verification. The number of \mathbb{G}_1 multi-EXPs, \mathbb{G}_T multi-EXPs and pairings are 22, 11 and 5 respectively. With precomputation, they become 14, 8 and 4. In addition to these calculation, Alice and Bob also need to compute several \mathbb{G}_1 multi-EXPs during the protocol.

Table 1 summarizes the computational costs for Alice and Bob.

7 Discussion

Resilience to Sybil Attacks. Sybil attacks [24] are attacks during which an individual entity masquerades as multiple simultaneous identities. Any authentication mechanisms including PPAA must defend Sybil attacks launched against

⁴ A multi-EXP computes the product of exponentiations faster than performing the exponentiations separately. We assume that one multi-EXP operation multiplies up to 3 exponentiations.

Table 1. Timing complexity of the Authentication protocol

	Number of Operations (without precomputation)		
	\mathbb{G}_1 multi-EXPs	\mathbb{G}_T multi-EXPs	Pairings
Alice (the Initiator)	12 (28)	4 (10)	2 (4)
Bob (the Responder)	16 (26)	8 (11)	4 (5)

user registration. Approaches exist to ensure that only legitimate users can register and that no legitimate user can register more than once. They include *trusted certification* such as X.509 [1], *resource testing*, where resources could be IP addresses or “friendship” in social networks or PGP-like web of trust, *recurring costs* imposed by cryptographic puzzles or CAPTCHAs [39], and *trusted devices* with certain degree of tamper-resistance, such as Trusted Platform Modules (TPMs) [35].

The practicality of the above approaches depends on the application scenarios. In the example of VANets, the Department of Motor Vehicles (DMV) can play the role of the GM with little overhead. Additionally, the makers of the vehicles can install a trusted device preloaded with a credential in each of vehicle they manufacture. In the example of P2P systems over a public network such the Internet, demonstrating the possession of IP addresses is a pragmatic and thus more popular approach, even though it does not have the highest resilience to Sybil attacks.

Revocation. Any practical authentication mechanism must allow for credential revocation. In the settings of PPAA, one might want to revoke a credential because the peer user in possession of that credential is compromised or misbehaving. For example, in VANets, the credential issued to a vehicle should be revoked when the vehicle is reported to have been stolen. Revocation allows for easier identification and thus tracking of stolen vehicles while maintaining the privacy of other vehicles as stolen cars with revoked credentials can no longer be anonymously authenticated by, e.g. a highway toll booth.

Our construction of PPAA can be modified in a straightforward manner to allow for credential revocation by adopting existing standard techniques [16, 11]: Alice and Bob verifiably encrypt part of their credentials during SPK_1 and SPK_2 respectively during the authentication under the public key of an entity usually referred to as the Revocation Manager. Now in addition to the original authentication, Alice and Bob have to convince one another that they have not been revoked. In the approach of verifier-local revocation [11], each user keeps a list of revoked users; in the approach of dynamic accumulators [16], each non-revoked user updates their credential when someone else’s has been revoked.

Authenticated Key-exchange. The authentication protocol in PPAA can be easily turned into an authenticated Diffie-Hellman key-exchange. Specifically, Alice additionally includes in m_1 an element g_0^a with $a \in_R \mathbb{Z}_p$ in Step 1 of the authentication protocol, while Bob additionally includes in m_2 an element g_0^b with $b \in_R \mathbb{Z}_p$ in Step 3. When the protocol terminates, both of them can derive a shared session

key as $g_0^{ab} = (g_0^a)^b = (g_0^b)^a$. Since m_1 and m_2 are signed with SPK_1 and SPK_2 respectively, Alice and Bob can use the session key to establish a confidential channel with the same privacy and accountability guarantees as in PPAA.

Blending Secret handshakes into PPAA. As discussed, anonymous SHSs such as Ateneise et al.’s [4] do not provide the linkability desired by the servers. On the other hand, PPAA leaks the initiating peer’s group affiliation to any responding peer who might not be a group member. Hence, each of them has its advantage over the other. Fortunately, one can enjoy the advantages of both by composing the two schemes. Specifically, two group members first execute an anonymous secret handshake to authenticate the group membership of one another and establish a secure channel, then they execute an PPAA authentication within that channel.

Furthermore, carrying out PPAA authentication within a secure channel has the additional benefit of preventing eavesdroppers from linking authentication traffic.

Fairness. In our PPAA construction, a malicious responding peer Bob might decide to stop after receiving Alice’s protocol message at step 3 of the authentication protocol so that he could learn Alice’s tag without Alice being able to learn his. The revealing of tags between Alice and Bob is thus not guaranteed to be fair in our construction.

Borrowing ideas from optimistic fair exchange [2, 3], one could augment fairness to PPAA by modifying it as follows. Alice requires Bob to additionally send a verifiable encryption of r_2 under the public key of some Trusted Third Party (TTP) in step 2 also that in case Bob stops before step 4, Alice can still reconstruct the tag with the help of the TTP. However, such a modification puts Bob’s privacy at risk, as the collusion between Alice the TTP can identify Bob. We leave the exploration of how to provide fairness without sacrificing privacy as future work.

8 Conclusion

In this paper, we have introduced *Peer-to-Peer Anonymous Authentication* (PPAA), a credential system that correctly balances user privacy and accountability in P2P systems where not just clients but also servers are concerned with their privacy. We have shown that such a credential system finds applications in many P2P systems such as VANets. We have presented the first PPAA construction, which is both secure and very efficient.

Acknowledgments

The authors would like to thank Man Ho Au for his suggestion on an initial tag design, the anonymous reviewers for their insightful reviews, and the members in the Security Reading Group at Dartmouth College (SRG@Dartmouth)⁵ who discussed this paper for their helpful feedback.

⁵ <https://wiki.cs.dartmouth.edu/srg/>

References

1. Adams, C., Farrell, S.: Internet X.509 Public Key Infrastructure Certificate Management Protocols. Internet Engineering Task Force: RFC 2510 (1999)
2. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: ACM Conference on Computer and Communications Security, pp. 7–17 (1997)
3. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures (extended abstract). In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
4. Ateniese, G., Blanton, M., Kirsch, J.: Secret Handshakes with Dynamic and Fuzzy Matching. In: NDSS, The Internet Society (2007)
5. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
6. Au, M.H., Susilo, W., Yiu, S.-M.: Event-oriented k -times revocable-iff-linked group signatures. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 223–234. Springer, Heidelberg (2006)
7. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.-C.: Secret Handshakes from Pairing-Based Key Agreements. In: IEEE Symposium on Security and Privacy, pp. 180–196. IEEE Computer Society Press, Los Alamitos (2003)
8. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and communications security, pp. 62–73. ACM Press, New York (1993)
9. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
10. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
11. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security, pp. 168–177. ACM Press, New York (2004)
12. Calandriello, G., Papadimitratos, P., Hubaux, J.-P., Liou, A.: Efficient and robust pseudonymous authentication in vanet. In: VANET 2007: Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks, pp. 19–28. ACM Press, New York (2007)
13. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n -times anonymous authentication. In: ACM Conference on Computer and Communications Security, pp. 201–210. ACM Press, New York (2006)
14. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
15. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
16. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)

17. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
18. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
19. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret handshakes from ca-oblivious encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
20. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO, pp. 199–203 (1982)
21. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
22. Christin, N., Weigend, A.S., Chuang, J.: Content availability, pollution and poisoning in file sharing peer-to-peer networks. In: ACM Conference on Electronic Commerce, pp. 68–77. ACM Press, New York (2005)
23. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
24. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
25. Galbraith, S.D., Rotger, V.: Easy Decision-Diffie-Hellman Groups. *LMS Journal of Computation and Mathematics* 7, 201–218 (2004)
26. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* 18(1), 186–208 (1989)
27. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* 17(2), 281–308 (1988)
28. Krumm, J.: Inference attacks on location tracks. In: LaMarca, A., Langheinrich, M., Truong, K.N. (eds.) Pervasive 2007. LNCS, vol. 4480, pp. 127–143. Springer, Heidelberg (2007)
29. Nguyen, L., Safavi-Naini, R.: Dynamic k-times anonymous authentication. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 318–333. Springer, Heidelberg (2005)
30. Raya, M., Hubaux, J.-P.: Securing vehicular ad hoc networks. *Journal of Computer Security* 15(1), 39–68 (2007)
31. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
32. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
33. Teranishi, I., Furukawa, J., Sako, K.: k-times anonymous authentication (extended abstract). In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 308–322. Springer, Heidelberg (2004)
34. Teranishi, I., Sako, K.: k-times anonymous authentication with a constant proving cost. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 525–542. Springer, Heidelberg (2006)
35. TPM Work Group. TCG TPM Specification Version 1.2 Revision 103. Technical report, Trusted Computing Group (2007)
36. Tsang, P.P., Smith, S.W.: PPAA: Peer-to-peer anonymous authentication (extended version). Technical Report TR2008-615, Department of Computer Science, Dartmouth College (April 2008)

37. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004)
38. Tsudik, G., Xu, S.: A flexible framework for secret handshakes. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 295–315. Springer, Heidelberg (2006)
39. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)
40. Xu, S., Yung, M.: k-anonymous secret handshakes with reusable credentials. In: ACM Conference on Computer and Communications Security, pp. 158–167. ACM Press, New York (2004)

Generic Constructions of Stateful Public Key Encryption and Their Applications

Joonsang Baek, Jianying Zhou, and Feng Bao

Institute for Infocomm Research

21 Heng Mui Keng Terrace

Singapore 119613

{jsbaek, jyzhou, baofeng}@i2r.a-star.edu.sg

Abstract. We present generic constructions of stateful public key encryption (StPE). We build several new StPE schemes and explain existing ones using our generic constructions. Of the new StPE schemes, two schemes are built using the “identity-based technique” whereby one can construct public key encryption (PKE) schemes secure against chosen ciphertext attack in the standard model from identity-based encryption (IBE) schemes. These StPE schemes provide a positive answer to Bellare et al.’s open question on whether stateful variants of PKE schemes derived from IBE schemes exist.

1 Introduction

The main goal of the stateful public key encryption (StPE) schemes proposed by Bellare, Kohno and Shoup [6] is to reduce the cost of public key encryption by allowing a sender to maintain *state* that is reused across different encryptions. For example, one can obtain a stateful version of the ElGamal encryption in which a message M is encrypted to $(g^r, g^{rx}M)$ for public key g^x by maintaining the random value r and its corresponding value g^r as state so that g^r does not need to be computed each time. (Note, however, that much more is involved in the analysis of this scheme.)

Reducing the computational cost of public key encryption is of particular importance for low-power mobile devices where computational resources are constrained (such as PDA and mobile phones) or sensors communicating with the relatively powerful servers or base stations [24,15,12]. Due to the efficiency gained from maintaining state, StPE schemes have potential to be employed in these settings. But, even in the environments that provide reasonable amount of computational resources, it is preferable to speed up public key operation, which is often more expensive than symmetric key operation, for overall system performance.

The approach that Bellare et al. [6] adopt to construct StPE schemes is to convert specific public key encryption schemes such as DHIES [1] and Kurosawa and Desmedt’s hybrid encryption scheme [20] into StPE schemes. However, for the practical reasons that speeding up public key operations is of great importance for the system performance, and new and more efficient computational

primitives may emerge in the future, it is desirable to have some generic methods to construct StPE schemes. – The application of our generic construction to Kiltz’s [18] new key encapsulation mechanism, which is presented in [4,3], is a good example of this argument.

Our Contributions. Regarding the issues discussed earlier we make the following contributions in this paper:

1. We formalize the concept of “partitioned” key encapsulation mechanism (PKEM) which is a special case of key encapsulation mechanism (KEM) [14] but turns out to encompass many existing schemes. Apart from the security against chosen ciphertext attack (IND-CCA) of KEM, we define some additional security properties that we require in our constructions of StPE. – See Section [2,2].
2. We present two generic constructions of StPE using PKEM and symmetric encryption. The first construction is shown to meet the strong security requirement defined in [6] in the known secret key (KSK) model without the random oracles [8]. The second construction is also shown to meet the security requirement in the unknown secret key (USK) model depending on the random oracles. – See Section [3].
3. We build several StPE schemes using the proposed generic constructions. Of these schemes, two are derived from the public key encryption (PKE) schemes constructed following the paradigm of converting identity-based encryption (IBE) into IND-CCA secure public key encryption (PKE) [13] in the *standard* model. We note that Bellare et al. [6] asked whether such StPE schemes exist. – See Section [4].

Related Work. Since Bellare et al. proposed the concept of StPE, to our knowledge, there has been few research work directly related to StPE. In their recent paper [25], Sarkar and Chatterjee discuss possible relation between the symmetric encryption they use for their generic construction of PKE from IBE and the symmetric encryption used in StPE. Other related work include the reuse of the randomness in the multi-receiver public key encryption, proposed by Kurosawa [19] and further formalized by Bellare, Boldyreva and Staddon [4]. (Readers are referred to [6] for detailed discussions on the relationship between StPE and the randomness reuse in the multi-receiver PKE.)

Organization of This Paper. In Section [2], we give definitions of all the building blocks we need in this paper. We then describe our generic constructions of StPE and give security analysis of them in Section [3]. In Section [4], we provide new StPE schemes derived from our generic constructions.

2 Building Blocks

2.1 Stateful Public Key Encryption

In this subsection we review the definitions of StPE and its security as given in [6].

Definition 1 (StPE). A stateful public key encryption scheme, denoted StPE, consists of the following algorithms:

- StPE.Setup: Taking 1^λ for a security parameter $\lambda \in \mathbb{Z}_{\geq 0}$ as input, this algorithm generates a system parameter sp which includes λ . We write $sp \leftarrow \text{StPE.Setup}(1^\lambda)$.
- StPE.KG: Taking sp as input, this algorithm generates a private/public key pair (sk, pk) . We write $(sk, pk) \leftarrow \text{StPE.KG}(sp)$.
- StPE.PKCK: Taking sp and pk as input, this algorithm returns 1 if the public key pk is valid and 0 otherwise. We write $\delta \leftarrow \text{StPE.PKCK}(sp, pk)$, where $\delta \in \{0, 1\}$.
- StPE.NwSt: Taking sp as input, this algorithm generates a new state. We write $st \leftarrow \text{StPE.NwSt}(sp)$.
- StPE.Enc: Taking sp, pk, st and a plaintext M as input, this algorithm outputs a ciphertext C and state st which may be different from the state provided as input to this algorithm. We write $(C, st) \leftarrow \text{StPE.Enc}(sp, pk, st, M)$.
- StPE.Dec: Taking sp, sk and C as input, this deterministic algorithm outputs M which is either a plaintext or \perp (meaning “reject”) message. We write $M \leftarrow \text{StPE.Dec}(sp, sk, C)$.

We impose a usual consistency condition on StPE: For any sp output by StPE.Setup, (sk, pk) generated by StPE.KG and st output by either StPE.NwSt or StPE.Enc, if (C, st) is an output of StPE.Enc(sp, pk, st, M), $\text{StPE.Dec}(sp, sk, C) = M$.

We remark that the state generated by StPE.NwSt algorithm, the state provided as input to the StPE.Enc algorithm and the state output by StPE.Enc algorithm can all be different from each other. Note that StPE.PKCK is a public key verification algorithm that checks the validity of the given public key. The level of the validity check we require in this paper is the same as that of the simple public key checking mechanisms, eg. checking whether some component of the given public key belongs to the underlying (mathematical) group, which are already exercised in practice [21, 17].

We now review the definition of chosen ciphertext security for StPE schemes as defined in [6].

Definition 2 (IND-CCA of StPE). Let StPE be a StPE scheme. Consider a game played with an attacker A :

Phase 1: The game computes $sp \leftarrow \text{StPE.Setup}(1^\lambda), (pk_1, sk_1) \leftarrow \text{StPE.KG}(sp)$ and $st \leftarrow \text{StPE.NwSt}(sp)$. (Note that (sk_1, pk_1) is the private/public key pair of the honest receiver R_1 .) The game sends (sp, pk_1) to A .

Phase 2: A outputs public keys pk_2, \dots, pk_n of receivers R_2, \dots, R_n respectively, all of which are in the range of $\text{KG}(sp)$. (Note that A may or may not know the private keys corresponding to the public keys pk_2, \dots, pk_n .)

Phase 3: A issues a number of (but polynomially many) queries, each of which is responded by the game. The type of each query and the action taken by the game are described as follows:

- A challenge query (M_0, M_1) such that $|M_0| = |M_1|$: The game picks $\beta \stackrel{\text{R}}{\leftarrow} \{0, 1\}$ (Throughout this paper, we denote by $s \stackrel{\text{R}}{\leftarrow} S$ the assignment of a uniformly and independently distributed random element from the set S to the variable s), computes $(C^*, st) \leftarrow \text{StPE.Enc}(sp, pk_1, st, M_\beta)$, where st denotes current state, and sends C^* to A .
- Encryption queries, each of which is denoted by (i, M) where $i \in \{1, \dots, n\}$: The game computes $(C, st) \leftarrow \text{StPE.Enc}(sp, pk_i, st, M)$, where st denotes current state, and sends C to A .
- Decryption queries, each of which is denoted by $C \neq C^*$: The game computes $\text{StPE.Dec}(sp, sk_1, C)$ and sends the resulting decapsulation (key or \perp (“Reject”)) to A .

Phase 4: A outputs its guess $\beta' \in \{0, 1\}$.

We define A 's advantage by $\text{Adv}_{A, \text{StPE}}^{\text{IND-CCA}}(\lambda) = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|$.

The chosen ciphertext security of StPE defined above can be considered in the *KSK* (Known Secret Key) or the *USK* (Unknown Secret Key) models [6]. In the KSK model, we assume that the attacker A possesses the corresponding private (secret) keys $sk_2 \dots, sk_n$ of the public keys it outputs in Phase 2 of the attack game. On the other hand, in the USK model, we do not need this assumption. – Namely, in the KSK model, it is likely that the CA (Certificate Authority) is required to perform a proof of knowledge protocol to confirm whether users have corresponding private keys of their public keys while in the USK model, StPE.PKCK should be run (by the game) to check whether the public keys are valid.

2.2 Partitioned Key Encapsulation Mechanism

In this subsection we define a new primitive called “partitioned key encapsulation mechanism (PKEM)” which is a special type of normal KEM [14]. Speaking informally, PKEM has a property that a part of ciphertext, which does not explicitly depend on the given public key (but depends on the system parameter as will be defined below), can be “partitioned” from other parts of ciphertext. Though this property seems somewhat special, we show in the later section that many KEM schemes are in fact PKEM.

Definition 3 (PKEM). A partitioned KEM scheme, which we simply denote by PKEM, consists of the following algorithms.

- PKEM.Setup: Taking 1^λ for a security parameter $\lambda \in \mathbb{Z}_{\geq 0}$ as input, this algorithm generates a system parameter sp which includes λ . sp also defines the key space \mathcal{K}_K . We write $sp \leftarrow \text{PKEM.Setup}(1^\lambda)$.
- PKEM.KG: Taking sp as input, this algorithm generates a private/public key pair (sk, pk) . We write $(sk, pk) \leftarrow \text{PKEM.KG}(sp)$.
- PKEM.Encap1 : Taking sp as input, this algorithm generates the first ciphertext vector ψ and state information ω which includes the internal randomness used to generate ψ . We write $(\omega, \psi) \leftarrow \text{PKEM.Encap1}(sp)$.

- PKEM.Encap2 : Taking sp, ω, pk and ψ as input, this algorithm generates the second ciphertext vector σ and a key K . We write $(\sigma, K) \leftarrow \text{PKEM.Encap2}(sp, pk, \omega, \psi)$.
- PKEM.Decap : Taking sp, sk, ψ and σ as input, this algorithm outputs either the session key K or the special symbol \perp . We write $K \leftarrow \text{PKEM.Decap}(sp, sk, \psi, \sigma)$.

We impose a consistency condition on PKEM: For any sp output by PKEM.Setup , (sk, pk) generated by PKEM.KG , if (ω, ψ) and (σ, K) are outputs of $\text{PKEM.Encap1}(sp)$ and $\text{PKEM.Encap2}(sp, pk, \omega, \psi)$ respectively, $\text{PKEM.Decap}(sp, sk, \psi, \sigma) = K$. We also impose the following *non-triviality* condition on (ψ, σ) and K : $\psi \neq \varepsilon$ and $K \neq \varepsilon$ but σ can be ε , where ε denotes empty string.

Note in the above definition that for the sake of convenience, we separate PKEM.Setup from the $\text{KEM.KeyGen}()$ algorithm given in [14] which generates both the system parameter and the public key together. Note also that the IND-CCA definition for PKEM is essentially no different from the usual IND-CCA definition for KEM [14] as an attacker does not get the state information ω during the attack. For completeness, however, we define IND-CCA of PKEMs.

Definition 4 (IND-CCA of PKEM). Let PKEM be a PKEM scheme. Consider a game played with an attacker A :

Phase 1: The game computes $sp \leftarrow \text{PKEM.Setup}(1^\lambda)$, $(pk, sk) \leftarrow \text{PKEM.KG}(sp)$ and gives (sp, pk) to A .

Phase 2: A issues decapsulation queries, each of which is denoted by (ψ, σ) . On receiving (ψ, σ) , the game computes $\text{PKEM.Decap}(sp, sk, \psi, \sigma)$ and gives the resulting decapsulation K (which can be \perp) to A .

Phase 3: The game subsequently computes $(\omega^*, \psi^*) \leftarrow \text{PKEM.Encap1}(sp)$ and $(\sigma^*, K_1^*) \leftarrow \text{PKEM.Encap2}(sp, pk, \omega^*, \psi^*)$. It also picks K_0^* at random from the key space \mathcal{K}_K . The game then picks $b \xleftarrow{R} \{0, 1\}$ and gives $(\psi^*, \sigma^*, K_b^*)$ to A .

Phase 4: A issues decapsulation queries, each of which is denoted by (ψ, σ) . A restriction here is that $(\psi, \sigma) \neq (\psi^*, \sigma^*)$. On receiving (ψ, σ) , the game computes

$\text{PKEM.Decap}(sp, sk, \psi)$ and gives the resulting decapsulation K (which can be \perp) to A . At the end of this phase, A outputs its guess $b' \in \{0, 1\}$.

We define A 's advantage by $\text{Adv}_{A, \text{PKEM}}^{\text{IND-CCA}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|$.

Proving the security of our generic construction of StPE in the KSK model given in Section 3.1 requires us to define a new property of PKEM, which we call “*reproducibility*”. Informally, this means that there exists a polynomial-time algorithm that, given a PKEM-ciphertext created under some public key and state information, and some other public/private key pair, produces another PKEM-ciphertext valid under the other public key and *the same state information* as the given ciphertext. (We note that a similar notion has been considered in [4] in the context of multi-receiver PKE.)

Intuitively, the reason why we require reproducibility is as follows. Given the specific state information, the adversary in the IND-CCA game of StPE (Definition 2) can come up with public keys of receivers other than the target receiver (denoted R_1) and produce ciphertexts associated with these public keys and the given state. In the KSK model, since the adversary is assumed to know private keys, each of which corresponds to each public key, he can produce such ciphertexts by himself. Now, a formal definition follows.

Definition 5 (Reproducibility of PKEM). Let PKEM be a PKEM scheme. Consider a game played with an algorithm R :

Phase 1: The game computes $sp \leftarrow \text{PKEM.Setup}(1^\lambda)$, $(sk, pk) \leftarrow \text{PKEM.KG}(sp)$, $(\omega, \psi) \leftarrow \text{PKEM.Encap1}(sp)$, $(\sigma, K) \leftarrow \text{PKEM.Encap2}(sp, pk, \omega, \psi)$ and $(sk', pk') \leftarrow \text{PKEM.KG}(sp)$. It gives $(sp, pk, \psi, \sigma, sk', pk')$ to R .

Phase 2: R outputs (σ', K') .

We define R 's advantage by

$$\mathbf{Adv}_{R, \text{PKEM}}^{\text{PKEM-Repr}}(\lambda) = \Pr[(\sigma', K') = \text{PKEM.Encap2}(sp, pk', \omega, \psi)].$$

We say that the PKEM scheme is reproducible if $\mathbf{Adv}_{R, \text{PKEM}}^{\text{PKEM-Repr}}(\lambda) = 1$.

Finally we define another type of security of PKEM, which is an extension of one-wayness (OW) under key checking attack (KCA) defined in [2]. (Note that KCA can be considered as a KEM version of the plaintext checking attack (PCA) defined in [23].) Our extension strengthens the OW-KCA of [2] in such a way that an attacker can freely choose a public key and include it to the “key checking” query. A formal definition, which we call “OW-EKCA (extended key checking attack)” is as follows.

Definition 6 (OW-EKCA of PKEM). Let PKEM be a PKEM scheme. Consider a game played with an attacker A :

Phase 1: The game computes $sp \leftarrow \text{PKEM.Setup}(1^\lambda)$, $(pk, sk) \leftarrow \text{PKEM.KG}(sp)$, $(\omega^*, \psi^*) \leftarrow \text{PKEM.Encap1}(sp)$ and $(\sigma^*, K^*) \leftarrow \text{PKEM.Encap2}(sp, pk, \omega^*, \psi^*)$, and gives $(sp, pk, \psi^*, \sigma^*)$ to A .

Phase 2: A issues key checking queries, each of which is denoted by $(pk', \psi', \sigma', K')$. On receiving it, the game checks whether (ψ', σ') encapsulates K' or not with respect to pk' . If it is, the game returns 1, and 0 otherwise. – We write this checking procedure as $\text{EKCO}(pk', \psi', \sigma', K')$, which returns 1 if (ψ', σ') encapsulates K' under the key pk' and 0 otherwise.

Phase 3: A outputs its guess K .

We define A 's advantage by $\mathbf{Adv}_{A, \text{PKEM}}^{\text{OW-EKCA}}(\lambda) = \Pr[K = K^*]$.

2.3 Symmetric Encryption

To construct IND-CCA secure StPE schemes, we need a somewhat strong symmetric encryption scheme. Note that in the usual KEM/DEM framework (DEM: Data Encapsulation Mechanism) for hybrid encryption [14], it is sufficient that the underlying symmetric encryption is IND-CCA in the weak sense that the attacker does not issue queries to the encryption oracle. In contrast, we need symmetric encryption secure against CCA attack in which the attacker does issue encryption queries. – A formal definition of IND-CCA for symmetric encryption can naturally be defined and can easily be found in the literature including [14].

We remark that as mentioned in [6], the symmetric encryption schemes meeting the IND-CCA definition can in fact be easily constructed, eg. using the encrypt-then-mac composition [7] with an AES mode of operation (such as CBC) and a MAC (such as CBC-MAC or HMAC [5]).

3 Our Constructions

3.1 Construction in the KSK Model

Description. We assume that a PKEM scheme PKEM and a symmetric encryption scheme SYM are “compatible” meaning that the key space \mathcal{K}_K of PKEM is the same as the key space \mathcal{K}_D of SYM. We use these schemes as building blocks to construct a stateful encryption scheme StPE. Below, we describe each sub-algorithm of StPE.

StPE.Setup is the same as PKEM.Setup, which outputs system parameter sp . Likewise, StPE.KG is the same as PKEM.KG, which outputs (sk, pk) , a private/public key pair. StPE.PKck simply returns 1 (and does nothing else) as the KSK model implies that any public keys in this system are generated correctly following the algorithm StPE.KG. (Namely the entity that has generated a public key must know the corresponding private key.)

In our construction of stateful encryption, we assume that only two types of state exist. The first type of state is produced by StPE.NwSt, which simply returns the output of PKEM.Encap1 on input sp . This state is kept unchanged until StPE.NwSt is invoked again to produce fresh state of the first type. The second type of state is produced by the algorithm StPE.Enc, which appends the first type of state output by StPE.NwSt to pk (provided as input to StPE.Enc) and the output of PKEM.Encap2. (Note here that PKEM.Encap2 takes (sp, pk) , the state output by StPE.NwSt and a plaintext M as input.) We also assume that pk and the output of PKEM.Encap2 of the second type of state is modified only by StPE.Enc. In what follows, we give algorithmic descriptions of StPE.NwSt, StPE.Enc and StPE.Dec.

Note that by the assumptions stated earlier, there are the following cases for st provided as input to StPE.Enc to become: 1) (ω, ψ) which is output of StPE.NwSt; 2) $(\omega, \psi, pk', \sigma', K')$ where σ' and K' are the outputs of PKEM.Encap2, both of which are created under the public key pk' different from the public key pk provided as input to StPE.Enc; and 3) $(\omega, \psi, pk, \sigma, K)$ where σ and K are created

under the public key pk provided as input to StPE.Enc . Note also that for state $st = (\omega, \psi)$ generated by the algorithm StPE.NwSt , $[\text{StPE.Enc}(sp, pk, st, M)]_C = [\text{StPE.Enc}(sp, pk, st', M)]_C$ for any st' output by StPE.Enc before StPE.NwSt is invoked to generate new state (different from st). Here, “ $[\text{StPE.Enc}(\dots)]_C$ ” denotes the ciphertext part of an output of StPE.Enc . – This property is used crucially to prove the security of the proposed construction.

```

StPE.NwSt(sp)
  ( $\omega, \psi$ )  $\leftarrow$  PKEM.Encap1(sp)
   $st \leftarrow (\omega, \psi)$ 
  Return  $st$ 
StPE.Enc(sp, pk, st, M)
  If  $st$  is of the form  $(\omega, \psi)$  or of the form  $(\omega, \psi, pk', \sigma', K')$  such that
   $pk' \neq pk$  then
    ( $\sigma, K$ )  $\leftarrow$  PKEM.Encap2(sp, pk,  $\omega, \psi$ )
  Else
    Parse  $st$  as  $(\omega, \psi, pk, \sigma, K)$ 
     $e \xleftarrow{R} \text{SYM.Enc}(K, M)$ 
     $C \leftarrow (\psi, \sigma, e)$ 
     $st \leftarrow (\omega, \psi, pk, \sigma, K)$ 
    Return  $(C, st)$ 
StPE.Dec(sp, sk, C)
  Parse  $C$  as  $(\psi, \sigma, e)$ 
   $K \leftarrow \text{PKEM.Decap}(sp, sk, \psi, \sigma)$ 
  If  $K = \perp$  then return  $\perp$ 
  Else return  $\text{SYM.Dec}(K, e)$ 

```

We remark that the StPE.Enc algorithm becomes *highly efficient* when a sender sends encryptions to a single receiver: If the sender wants to send encryptions of M_1, \dots, M_n to the same receiver whose public key is pk , he does not have to run PKEM.Encap2 and PKEM.Key for each plaintext M_i for $i = 1 \dots, n$ but just runs them once at the beginning and then only runs SYM.Enc on input (K, M_i) afterwards.

Security Analysis of Generic Construction. Our generic construction of StPE seems to be reminiscent of the KEM/DEM paradigm of constructing hybrid encryption given in [14]. However, the *PKEM-reproducibility* that we defined in the previous section (Definition 5) and the “reuse” of a ciphertext and its corresponding key of the PKEM scheme for the multiple encryptions without compromising confidentiality due to the strong security of the SYM scheme are the features that distinguish our generic construction of StPE from the KEM/DEM framework for constructing normal hybrid encryption. We now prove the following theorem.

Theorem 1. *In the KSK model, the proposed generic stateful public key encryption scheme StPE is IND-CCA secure if the underlying PKEM scheme PKEM is IND-CCA secure and reproducible, and the underlying symmetric encryption scheme SYM is IND-CCA secure. More precisely, we have*

$$\mathbf{Adv}_{A, \text{StPE}}^{\text{IND-CCA}}(\lambda) \leq 2\mathbf{Adv}_{B_1, \text{PKEM}}^{\text{IND-CCA}}(\lambda) + \mathbf{Adv}_{B_2, \text{SYM}}^{\text{IND-CCA}}(\lambda),$$

where λ denotes the security parameter; A , B_1 and B_2 denote the corresponding attackers.

Proof. The proof uses the technique of sequence of games [26].

- Game G_0 : This game is identical to the IND-CCA game played by an attacker A against StPE. (Readers are referred to Definition 2) We repeat this game to clean up the notations. Let sp be a system parameter. Let pk_1 and sk_1 be public and private keys of the honest receiver respectively. Let pk_2, \dots, pk_n be the public keys output by A . Let $st = (\omega^*, \psi^*)$, where $(\omega^*, \psi^*) \leftarrow \text{PKEM.Encap1}(sp)$, be the sender's state generated by StPE.NwSt , fixed throughout each game. We denote a challenge ciphertext by $C^* = (\psi^*, \sigma^*, e^*)$, where $(\sigma^*, K_1^*) \leftarrow \text{PKEM.Encap2}(sp, pk_1, \omega^*, \psi^*)$ and $e^* \stackrel{\text{R}}{\leftarrow} \text{SYM.Enc}(K_1^*, M_\beta)$ for $\beta \stackrel{\text{R}}{\leftarrow} \{0, 1\}$.

Now, observe that we can assume that A does not make encryption queries of the form (i, M) for $i = 2, \dots, n$. The reason is that since A is assumed to know sk_i corresponding to its public key pk_i following the KSK model, by the *reproducibility*, it, given (ψ^*, σ^*, pk_1) , can compute (σ_i, K_i) such that $(\sigma_i, K_i) \leftarrow \text{PKEM.Encap2}(sp, pk_i, \omega^*, \psi^*)$ for $i = 2, \dots, n$. Consequently, it can compute $e_i \stackrel{\text{R}}{\leftarrow} \text{SYM.Enc}(K_i, M)$ and can create ciphertext $C_i = (\psi^*, \sigma_i, e_i)$ for all $i = 2, \dots, n$.

We denote by S_0 the event $\beta' = \beta$, where β' is a bit output by A at the end of the game. (We use a similar notation S_1, S_2, \dots for all modified games G_1, G_2, \dots respectively). Since G_0 is the same as the real attack game of the IND-CCA of StPE, we have

- Game G_1 : In this game, we modify the generation of e^* (a component of challenge ciphertext) of the previous game as follows: $e^* \stackrel{\text{R}}{\leftarrow} \text{SYM.Enc}(K_0^*, M_\beta)$, where K_0^* is the key chosen at random from $\mathcal{K}_K (= \mathcal{K}_D)$ and $\beta \stackrel{\text{R}}{\leftarrow} \{0, 1\}$. Now, in the following, we construct an oracle machine B_1 that breaks IND-CCA of PKEM using A as a subroutine.

Algorithm $B_1(sp, pk_1)$

Give (sp, pk_1) to A

If A issues a challenge query (M_0, M_1) such that $|M_0| = |M_1|$ then

get a challenge ciphertext/key pair $(\psi^*, \sigma^*, K_b^*)$ where $b \stackrel{\text{R}}{\leftarrow} \{0, 1\}$

from the challenger; $\beta \stackrel{\text{R}}{\leftarrow} \{0, 1\}$; $e^* \stackrel{\text{R}}{\leftarrow} \text{SYM.Enc}(K_b^*, M_\beta)$;

$C^* \leftarrow (\psi^*, \sigma^*, e^*)$; Give C^* to A

If A issues an encryption query $(1, M)$ then

$e \stackrel{\text{R}}{\leftarrow} \text{SYM.Enc}(K_b^*, M)$; $C \leftarrow (\psi^*, \sigma^*, e)$; Give C to A .

If A issues a decryption query $C \neq C^*$, where $C = (\psi, \sigma, e)$, then

If $(\psi, \sigma) \neq (\psi^*, \sigma^*)$ then query (ψ, σ) to the challenger to get $K = \text{Decap}(sp, sk_1, \psi, \sigma)$

If $K \neq \perp$ then return $\text{SYM.Dec}(K, e)$

Else return \perp

Else return $\text{SYM.Dec}(K_b^*, e)$

If A outputs β' such that $\beta' = \beta$ then return $b' = 1$ (b' is B_1 's guess on b)

Else return $b' = 0$

First, assume that $b = 1$ in the above construction. In this case, note that K_1^* is the right key of PKEM. Importantly, note also that $[\text{StPE.Enc}(sp, pk, st, M)]_C = [\text{StPE.Enc}(sp, pk, st', M)]_C$ for any st' produced by StPE.Enc before StPE.NwSt is invoked to produce new state different from st . (Recall that “ $[\text{StPE.Enc}(\dots)]_C$ ” denotes the ciphertext part of an output of StPE.Enc .) Hence, the ciphertexts (ψ^*, σ^*, e^*) and (ψ^*, σ^*, e) provided as responses to A 's challenge and encryption queries respectively and those in the real attack game (which is Game G_0) are distributed identically. Decryptions are also perfectly simulated. Consequently, B_1 creates the same environment as Game G_0 in which A outputs its guess β' . Hence, we have $\Pr[S_0] = \Pr[\beta' = \beta] = \Pr[b' = 1|b = 1]$. Next, assume that $b = 0$ in the above construction. Note that in this case, B_1 creates the same environment as this game (Game G_1) in which K_0^* is the key chosen at random from $\mathcal{K}_K (= \mathcal{K}_D)$. Hence, we have $\Pr[S_1] = \Pr[\beta' = \beta] = \Pr[b' = 1|b = 0]$. Thus, we obtain

$$\begin{aligned} |\Pr[S_0] - \Pr[S_1]| &= |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]| \\ &= 2 \left(\left| \frac{1}{2} \Pr[b' = 1|b = 1] - \frac{1}{2} \Pr[b' = 1|b = 0] \right| \right) \\ &= 2 \left(\left| \frac{1}{2} \Pr[b' = 1|b = 1] + \frac{1}{2} (1 - \Pr[b' = 1|b = 0]) - \frac{1}{2} \right| \right) \\ &= 2 \left(\left| \frac{1}{2} \Pr[b' = 1|b = 1] + \frac{1}{2} \Pr[b' = 0|b = 0] - \frac{1}{2} \right| \right) \\ &= 2 \text{Adv}_{B_1, \text{PKEM}}^{\text{IND-CCA}}(\lambda). \end{aligned}$$

Now, in the following, we construct an oracle machine B_2 that breaks IND-CCA of SYM using the attacker A as a subroutine.

Algorithm $B_2(\lambda)$

Generate sp, sk_1 and pk_1 ; Give (sp, pk_1) to A

$(\omega^*, \psi^*) \leftarrow \text{PKEM.Encap1}(sp)$;

$(\sigma^*, K^*) \leftarrow \text{PKEM.Encap2}(sp, pk_i, \omega^*, \psi^*)$;

If A issues a challenge query (M_0, M_1) such that $|M_0| = |M_1|$ then

query (M_0, M_1) to the challenger to get $e^* \xleftarrow{R} \text{SYM.Enc}(K_0^*, M_\beta)$

where $\beta \xleftarrow{R} \{0, 1\}$; $C^* \leftarrow (\psi^*, \sigma^*, e^*)$; Give C^* to A .

If A issues an encryption query $(1, M)$ then

query M to the challenger to get $e \xleftarrow{R} \text{Enc}(K_0^*, M)$;

$C \leftarrow (\psi^*, \sigma^*, e)$; Give C to A .
 If A issues a decryption query $C \neq C^*$, where $C = (\psi, \sigma, e)$, then
 If $(\psi, \sigma) \neq (\psi^*, \sigma^*)$ then
 $K \leftarrow \text{PKEM.Decap}(sp, sk_1, \psi, \sigma)$
 If $K \neq \perp$ then return $\text{SYM.Dec}(K, e)$
 Else return \perp
 Else query e (which must be different from e^*) to the challenger to
 get $d = \text{SYM.Dec}(K_0^*, e)$; Return d
 If A outputs β' then return β'

Observe that in the above algorithm B_2 , A is essentially conducting chosen ciphertext attack on the symmetric encryption scheme SYM . Thus we have

$$\left| \Pr[S_1] - \frac{1}{2} \right| \leq \mathbf{Adv}_{B_2, \text{SYM}}^{\text{IND-CCA}}(\lambda).$$

3.2 Construction in the USK Model

Description. Let H be a random oracle [8], whose range (output-space) is the same as the key space \mathcal{K}_D of the symmetric encryption scheme SYM . Assume that there exists an algorithm PKV which checks whether public key of the given PKEM scheme PKEM is valid. We note that PKV is not an algorithm for “proving” the possession of the corresponding private key but a simple mechanism for validating keys or domain parameters by, eg. showing they belong to the output space of the key generation algorithm, as described in the public key cryptography standard such as P1363 [21]. (Readers are particularly referred to Section D.3.3 of the P1363 specification.)

Using the PKEM and SYM schemes and the random oracle [8] H as building blocks we construct another stateful encryption scheme StPE as follows. StPE.Setup is the same as PKEM.Setup , which outputs sp . Also, StPE.KG is the same as PKEM.KG , which outputs (sk, pk) . We assume here that sk includes pk . StPE.PKCh runs the algorithm PKV to check whether a given public key pk is in $\{\text{PKEM.KG}(sp)\}$.

Like the construction of stateful encryption in the KSK model presented in the previous section, we assume that there exist only two types of state. The first type of state is produced by StPE.NwSt , which simply returns the output of PKEM.Encap1 on input sp . This state is kept unchanged until StPE.NwSt is invoked again to produce fresh state of the first type. The algorithm StPE.Enc produces the second type of state by appending the first type of state output by StPE.NwSt to pk (provided as input to StPE.Enc) and the session key \tilde{K} output by H . (Note here that H takes as input the part of the state output by StPE.NwSt , pk and the output of PKEM.Encap2 .) We also assume that pk and the output of PKEM.Encap2 of the second type of state is modified only by StPE.Enc .

Security Analysis of Generic Construction. We now state the following theorem regarding the security of the construction presented above. – Due to the page limit, the proof will be provided in the full version of this paper.

```

StPE.NwSt(sp)
  (ω, ψ) ← PKEM.Encap1(sp)
  st ← (ω, ψ)
  Return st
StPE.Enc(sp, pk, st, M)
  If st is of the form (ω, ψ) or of the form (ω, ψ, pk', σ', K̃') such that
  pk' ≠ pk then
    (σ, K) ← PKEM.Encap2(sp, pk, ω, ψ)
    K̃ ← H(ψ, pk, σ, K)
  Else
    Parse st as (ω, ψ, pk, σ, K̃)
  e  $\xleftarrow{R}$  SYM.Enc(K̃, M)
  C ← (ψ, σ, e)
  st ← (ω, ψ, pk, σ, K̃)
  Return (C, st)
StPE.Dec(sp, sk, C)
  Parse C as (ψ, σ, e)
  K ← PKEM.Decap(sp, sk, ψ)
  If K = ⊥ then return ⊥
  Else
    K̃ ← H(ψ, pk, σ, K)
    Return SYM.Dec(K̃, e)

```

Theorem 2. *In the USK model, the proposed generic stateful public key encryption scheme StPE described above is IND-CCA secure if the underlying hash function H is modeled as random oracle; the underlying PKEM scheme is OW-EKCA secure; and the underlying SYM scheme is IND-CCA secure. More precisely, we have*

$$\mathbf{Adv}_{A, \text{StPE}}^{\text{IND-CCA}}(\lambda) \leq \mathbf{Adv}_{B_1, \text{PKEM}}^{\text{OW-EKCA}}(\lambda) + \mathbf{Adv}_{B_2, \text{SYM}}^{\text{IND-CCA}}(\lambda),$$

where λ denotes the security parameter; A , B_1 and B_2 denote the corresponding attackers as defined in Section 2.

4 Applications

4.1 A StPE Scheme Based on the Identity-Based Technique by Boyen, Mei and Waters

One of interesting applications of our generic constructions is to build StPE based on the *identity-based technique* which converts IBE schemes into (possibly efficient) IND-CCA secure PKE schemes in the standard model [13]. (Recall that Bellare et al. [6] asked whether the PKE schemes constructed in this way have stateful variants.) Among various identity-based techniques available in

the literature, we select, due to their efficiency, one of Boyen, Mei and Waters' presented in [11], and Boneh and Katz's one presented in [10], both of which are essentially yielded from the Boneh-Boyen selective-ID IBE [9].

We now describe a PKEM scheme derived from Boyen et al.'s KEM scheme in [11], which we denote by BMW. Readers are referred to Appendix 4.2 for the application of our generic construction of StPE to Boneh and Katz's identity-based technique [10].

Description of BMW PKEM. By simply rearranging Boyen et al.'s KEM scheme, one can obtain BMW as follows. On input 1^λ , BMW.Setup picks groups \mathbb{G} and $\hat{\mathbb{G}}$ of prime order q , generated by g and h respectively. It then constructs a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$. It picks a collision resistant hash function $H_s : \mathbb{G} \rightarrow \mathbb{Z}_q$. Finally it returns $sp = (\lambda, q, g, h, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, H_s)$. BMW.KG(sp) selects $\alpha \xleftarrow{\mathbb{R}} \mathbb{Z}_q$ and $l \leftarrow h^\alpha$, and computes $Z \leftarrow e(g, l)$. (Note that $l \in \hat{\mathbb{G}}$.) It then picks $x \xleftarrow{\mathbb{R}} \mathbb{Z}_q$ and $y \xleftarrow{\mathbb{R}} \mathbb{Z}_q$, and computes $u \leftarrow g^x$ and $v \leftarrow g^y$. It chooses a random seed s and returns $sk = (pk, l, x, y)$ and $pk = (s, Z, u, v)$. The rest of the algorithms are described as follows:

BMW.Encap1(sp)	BMW.Encap2(sp, pk, r, ψ)	BMW.Decap(sp, sk, ψ, σ)
$r \xleftarrow{\mathbb{R}} \mathbb{Z}_q; \psi \leftarrow g^r$	$w \leftarrow H_s(\psi)$	$w \leftarrow H_s(\psi)$
Return (r, ψ)	$\sigma \leftarrow u^r v^{rw}; K \leftarrow Z^r$	$\bar{w} \leftarrow x + yw \pmod{q}$
	Return (σ, K)	If $\psi^{\bar{w}} = \sigma$ then
		$K \leftarrow e(\psi, l);$ Return K
		Else return \perp

Note that in the above description r denotes state information (represented by ω in Definition 3). Hereinafter, we use r to denote state information.

StPE from BMW PKEM. In [11], the BMW scheme is shown to be IND-CCA secure assuming that the Decisional Bilinear Diffie-Hellman (DBDH) problem is intractable. We now prove that it satisfies reproducibility (Definition 5) as well.

Lemma 1. *BMW PKEM is reproducible.*

Proof. Let $sp = (\lambda, q, g, h, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, H)$ be a system parameter. Let $sk = (l, x, y)$ and $pk = (s, Z, u, v)$, where $l = h^\alpha$ for random $\alpha \in \mathbb{Z}_q$, $u = g^x$ and $v = g^y$, be private and public keys respectively. Suppose that another private/public key pair (sk', pk') such that $sk' = (l', x', y')$ and $pk' = (s', Z', u', v')$, where $l' = h^{\alpha'}$ for random $\alpha' \in \mathbb{Z}_q$, $u' = g^{x'}$ and $v' = g^{y'}$, is generated. Also, let $(\psi, \sigma) = (g^r, u^r v^{rw})$ for random $r \in \mathbb{Z}_q$, where $w = H_s(\psi)$, and $K = Z^r$.

Given $(sp, pk, \psi, \sigma, sk', pk')$, one can compute $w' \leftarrow H_{s'}(\psi); \sigma' \leftarrow \psi^{x'+y'w'}; K' \leftarrow e(\psi, l')$ and outputs (σ', K') . Note that $\psi^{w'} = g^{rw'} = g^{r(x'+y'w')} = \psi^{x'+y'w'} = \sigma'$ assuming that $\bar{w}' = x'+y'w' \pmod{q}$. Note also that $(g, \psi, u'v'^{w'}, \sigma')$ is a Diffie-Hellman tuple, and that $K' = e(\psi, l') = e(g, l')^r = Z'^r$. Thus, $(\sigma', K') = \text{PKEM.Encap2}(sp, pk', r, \psi)$.

Now, assume that the key space of \mathcal{K}_K of the BMW scheme is the same as that of the symmetric encryption scheme, which we denote by SYM. (Note that this can easily be achieved by providing the key K to the key derivation function (KDF) [14]. If the KDF is secure in the sense of “indistinguishability” as defined in [14] and the original KEM is IND-CCA, the resulting KEM scheme is also IND-CCA, which can easily be shown.) Then, if the SYM scheme is IND-CCA secure, by the result of Theorem 1, the StPE scheme based on BMW PKEM is IND-CCA secure in the KSK model without the random oracles.

4.2 A StPE Scheme Based on the Identity-Based Technique by Boneh and Katz

Another StPE scheme based on the identity-based technique can be built using our generic construction. This time the underlying PKEM scheme is derived from Boneh and Katz’s [10] identity-based technique. By BK, we denote this PKEM scheme.

Description of BK PKEM. On input 1^λ , BK.Setup first picks groups \mathbb{G} and \mathbb{G}_1 of prime order q , where \mathbb{G} is generated by g . It then constructs a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$. It selects a pseudorandom generator $G : \mathbb{G}_1 \rightarrow \{0, 1\}^*$, a second-preimage resistant hash function $H : \{0, 1\}^{448} \rightarrow \{0, 1\}^{128}$ and a message authentication scheme $\text{MAC}=(\mathcal{T}, \mathcal{V})$. \mathcal{T} and \mathcal{V} are tagging and verification algorithms respectively. Finally it returns $sp = (\lambda, q, g, \mathbb{G}, \mathbb{G}_1, e, e(g, g), G, H, \text{MAC})$. BK.KG(sp) picks $\alpha_1 \xleftarrow{R} \mathbb{Z}_q$, $\alpha_2 \xleftarrow{R} \mathbb{Z}_q$ and $x \xleftarrow{R} \mathbb{Z}_q$; computes $g_1 \leftarrow g^{\alpha_1}$, $g_2 \leftarrow g^{\alpha_2}$, $g_3 \leftarrow g^x$ and $Z \leftarrow e(g, g)^{\alpha_1 x}$. It chooses a hash function h from a family of pairwise independent hash functions. It returns $sk = (\alpha_1, \alpha_2, x, h)$ and $pk = (g_1, g_2, g_3, Z, h)$. The rest of the algorithms are described as follows:

BK.Encap1(sp)	BK.Encap2(sp, pk, r, ψ)	BK.Decap(sp, sk, ψ, σ)
$r \xleftarrow{R} \mathbb{Z}_q; \psi \leftarrow g^r$	$s \xleftarrow{R} \{0, 1\}^{448}$	Parse σ as $(\rho, \theta, \phi, \tau)$;
Return (r, ψ)	$k_1 \leftarrow h(s); K \xleftarrow{R} \mathcal{K}_K$	$t \xleftarrow{R} \mathbb{Z}_q; (K s) \leftarrow \phi \oplus$
	$\rho \leftarrow H(s); \theta \leftarrow g_2^r g_3^{\rho}$	$G(e(\psi^{\alpha_1 x + t(\alpha_2 + x\rho)} \theta^{-t}, g))$
	$\phi \leftarrow G(Z^r) \oplus (K s)$	$k_1 \leftarrow h(s)$
	$\tau \leftarrow \mathcal{T}(k_1, (\psi, \theta, \phi))$	If $\mathcal{V}(k_1, (\psi, \theta, \phi), \tau) = 1$ and
	$\sigma \leftarrow (\rho, \theta, \phi, \tau)$	$H(s) = \rho$ then
	Return (σ, K)	return K
		Else return \perp

StPE from BK KEM. We first prove that BK has the reproducibility (Definition 5).

Lemma 2. *BK PKEM is reproducible.*

Proof. Let $sp = (\lambda, q, g, \mathbb{G}, \mathbb{G}_1, e, e(g, g), G, H, \text{MAC})$ be a system parameter as defined earlier. Let $sk = (\alpha_1, \alpha_2, x, h)$ and $pk = (g_1, g_2, g_3, Z, h)$, where $g_1 = g^{\alpha_1}$, $g_2 = g^{\alpha_2}$, $g_3 = g^x$, $Z = e(g, g)^{\alpha_1 x}$ and h is drawn from a family of pairwise

independent hash functions, be private and public keys respectively. Suppose that another private/public key pair (sk', pk') such that $sk' = (\alpha'_1, \alpha'_2, x', h')$ and $pk' = (g'_1, g'_2, g'_3, Z', h')$, where $g'_1 = g^{\alpha'_1}$, $g'_2 = g^{\alpha'_2}$, $g'_3 = g^{x'}$, $Z' = e(g, g)^{\alpha'_1 x'}$ and h' is drawn from a family of pairwise independent hash functions, is generated. Also, let $\psi = g^r$ and $\sigma = (\rho, \theta, \phi, \tau)$, where $\rho = H(s)$, $\theta = g_2^r g_3^{r\rho}$, $\phi = G(Z^r) \oplus (K||s)$ and $\tau = \mathcal{T}(k_1, (\psi, \theta, \phi))$, where $k_1 = h(s)$, for random $r \in \mathbb{Z}_q$, $s \in \{0, 1\}^{448}$ and $K \in \mathcal{K}_K$.

Given $(sp, pk, \psi, \sigma, sk', pk')$, one can compute $s' \xleftarrow{R} \{0, 1\}^{448}$; $\rho' \leftarrow H(s')$; $\theta' \leftarrow \psi^{\alpha'_2} \psi^{x' \rho'}$; $K' \xleftarrow{R} \mathcal{K}_K$; $\phi' \leftarrow G(e(\psi, g)^{\alpha'_1 x'}) \oplus (K' || s')$; $k'_1 \leftarrow h'(s')$; $\tau' \leftarrow \mathcal{T}(k'_1, (\psi, \theta', \phi'))$ and output $\sigma' = (\rho', \theta', \tau')$ and K' .

Note that $\theta' = \psi^{\alpha'_2} \psi^{x' \rho'} = g^{r\alpha'_2} g^{r x' H(s')} = (g_2^r)^{\alpha'_2} (g_3^r)^{H(s')}$. Note also that $e(\psi, g_1^{\alpha'_1})^{x'} = e(g, g_1^{\alpha'_1})^{r x'} = Z'^r$. It is clear that τ' is valid. Thus, $(\sigma', K') = \text{BK.Encap2}(sp, pk', r, \psi)$.

Note that the above BK PKEM scheme is derived simply from the Boneh and Katz's PKE scheme (converted from selective-ID IBE) by providing a random key instead of a plaintext as input to the encryption algorithm (and separating the ciphertext part which depends on the system parameter only). One can easily show that if Boneh and Katz's PKE scheme is IND-CCA secure relative to the DBDH problem, which is actually shown in [10], the BK KEM scheme is also IND-CCA secure assuming that the DBDH problem is hard. Now, assume that the key space of \mathcal{K}_K of the BK scheme is the same as that of the symmetric encryption scheme SYM. Then, if the SYM scheme is IND-CCA secure, by the result of Theorem 1 the StPE scheme from BK PKEM is IND-CCA secure in the KSK model without random oracles.

4.3 A StPE Scheme from Kiltz's KEM Scheme

One can also apply the generic construction presented in Section 3.1 to the KEM scheme proposed by Kiltz [18] very recently. Next, we describe the PKEM version of this scheme, which we denote by Kl.

Description of Kl PKEM. On input 1^λ , Kl.Setup picks a group \mathbb{G} of prime order q , generated by g . It then picks a target-collision resistant hash function $H : \mathbb{G} \rightarrow \mathbb{Z}_q$ and a key derivation function KDF. Finally it returns $sp = (\lambda, q, g, \mathbb{G}, H, \text{KDF})$. Kl.KG(sp) picks $x \xleftarrow{R} \mathbb{Z}_q$ and $y \xleftarrow{R} \mathbb{Z}_q$, and computes $u \leftarrow g^x$ and $v \leftarrow g^y$. It returns $sk = (x, y)$ and $pk = (u, v)$. The rest of the algorithms are described as follows:

Kl.Encap1(sp)	Kl.Encap2(sp, pk, r, ψ)	Kl.Decap(sp, sk, ψ, σ)
$r \xleftarrow{R} \mathbb{Z}_q^*$; $\psi \leftarrow g^r$; Return (r, ψ)	$t \leftarrow H(\psi)$; $\sigma \leftarrow (u^t v)^r$ $K \leftarrow \text{KDF}(u^r)$ Return (σ, K)	$t \leftarrow H(\psi)$ $K \leftarrow \text{KDF}(\psi^x)$ If $\psi^{x+t+y} = \sigma$ Return K Else return \perp

StPE from KI PKEM. We prove that the above KI PKEM scheme satisfies the reproducibility.

Lemma 3. *KI PKEM is reproducible.*

Proof. Let $sp = (\lambda, q, g, \mathbb{G}, H, \text{KDF})$ be a system parameter. Let $sk = (x, y)$ and $pk = (u, v)$, where $u = g^x$ and $v = g^y$. Suppose that another private/public key pair (sk', pk') such that $sk' = (x', y')$ and $pk' = (u', v')$, where $u' = g^{x'}$ and $v' = g^{y'}$, is generated. Also, let $(\psi, \sigma) = (g^r, (u^t v)^r)$, where $t = H(\psi)$.

Given $(sp, pk, \psi, \sigma, sk', pk')$, one can compute $\sigma' \leftarrow \psi^{x't+y'}$; $K' \leftarrow \text{KDF}(\psi^{x'})$, where $t = H(\psi)$, and output (σ', K') . Note that $\sigma' = g^{r(x't+y')} = (g^{x't} g^{y'})^r = (u'^t v')^r$. Thus, $(\sigma', K') = \text{PKEM.Encap2}(sp, pk', r, \psi)$.

It is shown in [18] that the above KI PKEM scheme is IND-CCA secure assuming that the Gap Hashed Diffie-Hellman (GHDH) problem is hard and the underlying hash function H is target-collision resistant.

Thus, assuming that the key space of \mathcal{K}_K of the KD scheme is the same as that of the symmetric encryption scheme SYM and the SYM scheme is IND-CCA secure, the StPE scheme based on KI PKEM is IND-CCA secure in the KSK model without the random oracles, by the result of Theorem [1].

4.4 A StPE Scheme from the Diffie-Hellman KEM Scheme

The stateful version of DHIES [1] proposed in [6] can actually be explained using our generic construction presented in Section 3.2. We now present a PKEM version of the Diffie-Hellman KEM, which we denote by DH as follows.

Description of DH PKEM. On input 1^λ , DH.Setup picks a group \mathbb{G} of prime order q , generated by g . It then returns $sp = (\lambda, q, g, \mathbb{G})$. $\text{DH.KG}(sp)$ picks $x \xleftarrow{\mathbb{R}} \mathbb{Z}_q$ and computes $y \leftarrow g^x$. It returns $sk = x$ and $pk = y$. The rest of the algorithms are described as follows:

$\text{DH.Encap1}(sp)$	$\text{DH.Encap2}(sp, pk, r, \psi)$	$\text{DH.Decap}(sp, sk, \psi, \sigma)$
$r \xleftarrow{\mathbb{R}} \mathbb{Z}_q^*$; $\psi \leftarrow g^r$	$\sigma \leftarrow \varepsilon$ (empty string)	$K \leftarrow \psi^x$
Return (r, ψ)	$K \leftarrow y^r$	Return K
	Return (σ, K)	

StPE from DH PKEM. We prove that the above DH PKEM scheme is OW-EKCA secure (Definition [6]).

Lemma 4. *DH PKEM is OW-EKCA secure assuming that Gap Diffie-Hellman (GDH) problem [22] is intractable.*

Proof. Assume that an GDH attacker B is given an instance $(\lambda, q, \mathbb{G}, g, g^a, g^b)$. B sets $sp = (\lambda, q, g, \mathbb{G})$, $pk = y = g^b$, $\psi^* = g^a$ and $\sigma^* = \varepsilon$ (empty string). B gives $(sp, pk, \psi^*, \sigma^*)$ to an OW-EKCA attacker A . Whenever A issues an EKCA query $(pk', \psi', \sigma', K')$, B forwards the query to its DDH oracle and sends back the

response $\text{DDH}_g(pk', \psi', K')$ to A . (Note here that the DDH oracle $\text{DDH}_g(\cdot, \cdot, \cdot)$ returns 1 if a given tuple is (g^u, g^v, g^{uv}) for $u, v \in \mathbb{Z}_q^*$ and returns 0 otherwise.) When A outputs it guess \bar{K} , B returns it.

Hence, assuming that the key space of \mathcal{K}_K of the DH scheme is the same as that of the symmetric encryption scheme SYM and the SYM scheme is IND-CCA secure, the StPE scheme based on DH is IND-CCA secure in the USK and random oracle model, by the result of Theorem 2.

5 Discussions

It is clear that the StPE schemes built from BMW PKEM in the Section 4.1 and BK PKEM in Appendix 4.2 are more efficient than the original PKE schemes based on the identity-based techniques presented in [11] and [10] respectively since the PKEM ciphertext (ψ, σ) and the key K are *reused* across encryptions directed to one receiver whose public key is pk . The proven security of these schemes ensures that reusing one instance of internal randomness appeared in the “state” across multiple receivers do not compromise the confidentiality. Moreover, the proof of security does not depend on the random oracles.

We note that the stateful version of the Kurosawa-Desmedt PKE scheme presented in [6] cannot be explained using our generic construction since the underlying PKEM defined in the same way as [20] is not IND-CCA secure as shown in [16]. However, we remark that by defining an extension of our approach based PKEM called “Tag-PKEM” (similar to Tag-KEM [3]), which provides a tag as input to PKEM.Encap2 , one could analyze the stateful version of the Kurosawa-Desmedt PKE scheme in [6]. But we realize that a definition of IND-CCA of Tag-PKEM needs to allow an attacker to have access to a new kind of encapsulation oracle which returns outputs of PKEM.Encap2 computed under the *same internal state* but different tags. Note that this oracle is required to simulate the encryption oracle of StPE, when it is queried by $(1, M_1), \dots, (1, M_n)$, whose responses should be encryptions of M_1, \dots, M_n under the same state. (In the IND-CCA definition of *stateless* PKE, this special type of oracle is not required.) Consequently, one cannot “reuse” the results of the security analysis of various Tag-KEMs available in the literature, as they need to be analyzed under this new security definition which is stronger than the normal IND-CCA definition of Tag-KEM given in [3].

Finally, we remark that along with the efficiency issue pointed out in [6], there is also a technical reason why constructing stateful versions of the RSA (or possibly usual integer factorization based schemes) is not very feasible: The reproducibility of PKEM (Definition 5) or OW-EKCA (Definition 6) is indeed strong property that many integer factorization based encryption schemes in which different receivers should have different modulus to guarantee security are not likely to satisfy. Nevertheless, more elaboration on this issues would be an interesting area of research.

Acknowledgement

The authors are grateful to the anonymous referees of ACNS '08 for their helpful comments. This work is partially funded by the European Union project SMEPP-033563.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Abe, M.: Combining Encryption and Proof of Knowledge in the Random Oracle Model. *Comput. J.* 47(1), 58–70 (2004)
3. Abe, M., Genaro, R., Kurosawa, K.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM, *Cryptology ePrint Archive: Report*, 2005/027 (2005) (This is the full version of their Eurocrypt 2005 paper with the same title)
4. Bellare, M., Boldyreva, A., Staddon, J.: Randomness Re-use in Multi-recipient Encryption Schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2002)
5. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
6. Bellare, M., Kohno, T., Shoup, V.: Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation. In: ACM-CCS 2006, pp. 380–389. ACM Press, New York (2006)
7. Bellare, M., Namprepre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
8. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM-CCS 1993, pp. 62–73. ACM Press, New York (1993)
9. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
10. Boneh, D., Katz, J.: Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
11. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-Based Techniques. In: ACM-CCS 2005, pp. 320–329. ACM Press, New York (2005)
12. Bresson, E., Chevassut, O., Essiari, A., Pointcheval, D.: Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In: IFIP-TC6 International Conference on Mobile and Wireless Communications Networks, pp. 59–62. World Scientific Publishing, Singapore (2003)
13. Canetti, R., Halevi, S., Katz, J.: Chosen Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
14. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal of Computing* 33, 167–226 (2003)

15. Gaubatz, G., Kaps, J.-P., Sunar, B.: Public Key Cryptography in Sensor Networks Revisited. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313. Springer, Heidelberg (2005)
16. Herranz, J., Hofheinz, D., Kiltz, E.: The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure, Cryptology ePrint Archive, Report 2006/207 (2006)
17. ISO 18033-2, An Emerging Standard for Public-Key Encryption (2004)
18. Kiltz, E.: Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
19. Kurosawa, K.: Multi-recipient Public-Key Encryption with Shortened Ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
20. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
21. IEEE P1363, Standard Specifications For Public-Key Cryptography (2000)
22. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
23. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
24. Phan, T., Huang, L., Dulun, C.: Challenge: Integrating Mobile Wireless Devices Into the Computational Grid. In: MobiCom 2002, pp. 271–278. ACM Press, New York (2002)
25. Sarkar, P., Chatterjee, S.: New Generic Constructions of Public Key Encryption from Identity Based Encryption, Cryptology ePrint Archive: Report 2007/067 (2007)
26. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs, Cryptology ePrint Archive: Report 2004/332 (2004)

Traceable and Retrievable Identity-Based Encryption*

Man Ho Au¹, Qiong Huang², Joseph K. Liu³, Willy Susilo¹, Duncan S. Wong²,
and Guomin Yang²

¹ Centre for Computer and Information Security Research (CCISR)
School of Computer Science and Software Engineering,
University of Wollongong, Australia
{mhaa456, wsusilo}@uow.edu.au

² Department of Computer Science, City University of Hong Kong, Hong Kong
csqhuang@student.cityu.edu.hk,

{duncan, csyanggm}@cs.cityu.edu.hk

³ Cryptography and Security Department
Institute for Infocomm Research, Singapore
ksliu@i2r.a-star.edu.sg

Abstract. Very recently, the concept of Traceable Identity-based Encryption (IBE) scheme (or Accountable Authority Identity based Encryption scheme) was introduced in Crypto 2007. This concept enables some mechanisms to reduce the trust of a private key generator (PKG) in an IBE system. The aim of this paper is threefold. First, we discuss some subtleties in the first traceable IBE scheme in the Crypto 2007 paper. Second, we present an extension to this work by having the PKG's master secret key retrieved automatically if more than one user secret key are released. This way, the user can produce a concrete *proof of misbehaviour* of the PKG in the court. In contrast to previous approach, our idea gives strong incentive for the PKG to strengthen the security of the system since if someone can successfully release a user's secret key, it means that his security is also compromised. We present a formal model to capture our idea. Third, we present an efficient construction based on Gentry's IBE that satisfies our model and prove its security. Our construction is proven secure in the random oracle model. Nevertheless, we should emphasize that the aim of this paper is to introduce the new model to strengthen the IBE system.

Keywords: Identity-based Encryption, Traceability, Retrievability, PKG, Trust.

1 Introduction

The idea of identity-based encryption (IBE) was put forward by Shamir in his seminal paper in [1]. The main concept was proposed in 1984, whilst the first

* M. H. Au and W. Susilo's work was partially supported by ARC Linkage Project LP0667899 and ARC Discovery Grants DP0663306 and DP0877123. Q. Huang, D. S. Wong and G. Yang's work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (RGC Ref. No. CityU 122107).

practical and fully functional IBE scheme was only proposed in [2] that takes advantage of the properties of suitable bilinear maps (the Weil or Tate pairing) over supersingular elliptic curves. Since then, many new schemes have been proposed in the literature (including the recent one by Gentry [3]). The main essence of IBE is to remove the necessity of public key certification, that is required in the conventional public key cryptography setting. The public key of each participant is obtained from his/her public identity, such as email address, IP address combined with a user name, social security number, etc. that can uniquely identify the participant. Furthermore, the sender can send his ciphertext to a recipient without requiring the receiver to have his public key setup first. Indeed, the secret key can be retrieved later after the receiver receives the ciphertext sent by the sender. Unfortunately, this model requires the existence of a *trusted* authority called the Private Key Generator (PKG), whose task is to generate user's private key from their identity information, after a successful identification.

In an IBE system, the PKG is completely trusted, and therefore the PKG has the liberty to engage in any malicious activity without any risk of being sent to court. To mitigate this trust problem, a distributed PKG was proposed [2]. Very recently, Goyal [4] presented a new idea to reduce the trust of the PKG. His idea is to produce an exponential (or super-polynomial) number of possible decryption keys corresponding to every identity. The PKG does not know which secret key that has been chosen by the user. Therefore, when the PKG releases one of the possible user's secret keys, then the user can later show two different secret keys as his proof of the PKG misbehaviour. Goyal formalized this notion as a *traceable identity-based encryption* scheme [4] (This notion was renamed to *Accountable Authority Identity-based Encryption* (A-IBE) in [5]).

Nonetheless, we believe that traceable IBE is *not* very useful for achieving the purpose of deterring the PKG from distributing private keys for any identity. The reason is that in practice, it is difficult for a user to win a court if the user sues the PKG. This is because the PKG can always put a disclaimer well in advance for mitigating the liability of the PKG. Another reason is that the damage is externality with respect to the PKG, rather than the PKG itself. Therefore, there is no strong incentive for the PKG to secure its own system. We therefore motivate ourselves with an additional mechanism which can help discourage the PKG from distributing private keys for any identity while encouraging the PKG to strengthen the security of its own system.

Our Contributions

We take one step forward than Goyal's idea in reducing the trust on the PKG. Our idea is to have the PKG's master secret key retrieved automatically if more than one user secret key are released. This way, the user can produce a concrete *proof of misbehaviour* of the PKG in the court. In contrast to Goyal's approach, our idea also gives some benefit to the PKG to strengthen their security system as if someone can successfully release a user's secret key, it means that his security

is also compromised. Therefore, it is also the PKG's interest to ensure its security of the system (c.f. Goyal's approach [4]).

In this paper, firstly we point out some subtleties in Goyal's work [4, 5]. More specifically, there are some subtleties in instantiating the ZK-POK sub-protocol in the first traceable IBE scheme provided in [4, 5] and we propose a suggestion on how to efficiently instantiate it. Furthermore, we observe that several definitions used in Goyal's work are *not* formally defined. Hence, the model provided in Goyal's work is incomplete. We present a formal model to capture our idea mentioned above with the aim of reducing the trust on the PKG. We deal with this part in two stages. Firstly, we formally define the parts that are lacking from Goyal's work. Then, we add an algorithm called `Retrieve` that is used to output the master secret key given two different user's secret keys. We call this notion as a *Traceable and Retrievable Identity-based Encryption scheme*. Second, we present an efficient construction based on Gentry's IBE [3] that satisfies our model.

Paper Organization

The rest of this paper is organized as follows. In Sec. 3, we present some security remarks on Goyal's work [4, 5]. In Sec. 4, we present the formal model of Traceable and Retrievable IBE. We present a concrete construction based on Gentry's IBE in Sec. 5. Finally, we conclude the paper in Sec. 6.

2 Preliminaries

Notations

Let e be a bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$.

- \mathbb{G}_1 and \mathbb{G}_2 are cyclic multiplicative groups of prime order p .
- each element of \mathbb{G}_1 , \mathbb{G}_2 and \mathbb{G}_3 has unique binary representation.
- g, h are generators of \mathbb{G}_1 and \mathbb{G}_2 respectively.
- (Bilinear) $\forall x \in \mathbb{G}_1, y \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p, e(x^a, y^b) = e(x, y)^{ab}$.
- (Non-degenerate) $e(g, h) \neq 1$.

\mathbb{G}_1 and \mathbb{G}_2 can be the same or different groups. We say that two groups ($\mathbb{G}_1, \mathbb{G}_2$) are a bilinear group pair if the group action in $\mathbb{G}_1, \mathbb{G}_2$ and the bilinear mapping e are all efficiently computable.

Complexity Assumptions

The security of our concrete construction is based on a complexity assumption called "truncated decision q -ABDHE assumption" proposed in [3] which is defined as follows:

Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map, where \mathbb{G} and \mathbb{G}_T are cyclic groups of large prime order p . Given a vector of $q + 3$ elements:

$$(g', g'^{(\alpha)^{q+2}}, g, g^\alpha, g^{(\alpha)^2}, \dots, g^{(\alpha)^q}) \in \mathbb{G}^{q+3}$$

and an element $Z \in \mathbb{G}_T$ as input, output 0 if $Z = e(g^{(\alpha)^{q+1}}, g')$ and output 1 otherwise.

An algorithm \mathcal{B} has advantage ϵ in solving the truncated decision q -ABDHE if:

$$\left| \Pr[\mathcal{B}(g', g'^{(\alpha)^{q+2}}, g, g^\alpha, \dots, g^{(\alpha)^q}, e(g^{(\alpha)^{q+1}}, g')) = 0] - \Pr[\mathcal{B}(g', g'^{(\alpha)^{q+2}}, g, g^\alpha, \dots, g^{(\alpha)^q}, Z) = 0] \right| \geq \epsilon$$

where the probability is over the random choice of generators g, g' in \mathbb{G} , the random choice of α in \mathbb{Z}_p , the random choice of Z in \mathbb{G}_T , and the random bits consumed by \mathcal{B} .

The computational version of the assumption is defined in the natural way, where the term Z is asked as the output.

Definition 1. We say that the truncated decision (t, ϵ, q) -ABDHE assumption holds in \mathbb{G} if no t -time algorithm has advantage at least ϵ in solving the truncated decision q -ABDHE problem in \mathbb{G} .

3 On Goyal's Scheme [4]

In this section we analyze some subtleties in Goyal's paper [4]. These subtleties are mainly about the instantiation of ZK-POK sub-protocol and the *FindKey* game. We also make some comments on Goyal's definition of Traceable IBE. First of all, we review Goyal's first traceable IBE scheme below.

3.1 Review of Goyal's First Traceable IBE Scheme

Goyal's first scheme [4] is built on top of Gentry's IBE scheme [3]. The basic cryptosystem (Setup, Encryption and Decryption) are taken from Gentry's scheme [3]. The only difference between Goyal's scheme and Gentry's scheme relies on the *Key Generation Protocol*, which is an interactive protocol between a user U and the PKG. For completeness, we review the *Setup* and *Key Generation Protocol* as follows.

Let \mathbb{G} be a bilinear group of large prime order p and let g be a generator of \mathbb{G} . Additionally, let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ denote a bilinear map. A security parameter, k , will determine the size of the groups.

Setup. The PKG picks random generators $g, h_1, h_2, h_3 \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It sets $g_1 = g^\alpha$ and then selects a hash function H from a family of universal one-way hash function. The published public parameters PK and the master key MK are given by

$$PK = \{g, g_1, h_1, h_2, h_3, H\}, \quad MK = \alpha$$

Key Generation Protocol. This protocol will allow a user U to securely obtain a decryption key d_{ID} from PKG. As in [3], PKG aborts if $\text{ID} = \alpha$. The key generation protocol is as follows.

1. The user U selects a random $r \in \mathbb{Z}_p$ and sends $R = h_1^r$ to the PKG.
2. U gives to PKG a zero-knowledge proof of knowledge of the discrete log of R with respect to h_1 .
3. The PKG now selects three random numbers $r', r_{ID,2}, r_{ID,3} \in \mathbb{Z}_p$. It then computes $h'_{ID,1} = (Rg^{-r'})^{1/(\alpha-ID)}$ and $h_{ID,i} = (h_i g^{-r_{ID,i}})^{1/(\alpha-ID)}$, $i \in \{2, 3\}$ and sends

$$\{r', h'_{ID,1}, r_{ID,2}, h_{ID,2}, r_{ID,3}, h_{ID,3}\}$$

to the user U .

4. U computes $r_{ID,1} = r'/r$ and $h_{ID,1} = (h'_{ID,1})^{1/r}$. It sets the decryption key $d_{ID} = \{(r_{ID,i}, h_{ID,i}) : i \in \{1, 2, 3\}\}$.
5. U now runs a key sanity check on d_{ID} as follows. It computes $g^{\alpha-ID} = g_1/g^{ID}$ and checks if $e(h_{ID,i}, g^{\alpha-ID}) \stackrel{?}{=} e(h_i g^{-r_{ID,i}}, g)$ for $i \in \{1, 2, 3\}$. U aborts if the check fails for any i .

At the end of this protocol, U will have a well-formed decryption key d_{ID} for the identity ID.

3.2 Comments on the Instantiation of ZK-POK

In the above scheme, a user runs a *Key Generation Protocol*, which is a zero knowledge proof of knowledge (ZK-POK), with the PKG to jointly generate his/her secret key, without letting the PKG know which key was actually generated. In this protocol, the user first randomly selects R and then proves to the PKG that he/she knows the discrete log of R with respect to base h_1 . We believe that the *Key Generation Protocol* is correct. However, there are some subtleties in instantiating the ZK-POK sub-protocol. Goyal suggested to employ Schnorr’s 3-round identification scheme [6] as the underlying ZK-POK, which is an *honest-verifier* zero-knowledge proof of knowledge. The revised and extended version [5] does not discuss much about the instantiation either. Proof systems proposed in [7] may not fit the *Key Generation Protocol*, as they merely concentrate on honest verifiers as well. It turns out that efficient instantiations of ZK-POK is not a very trivial task as one may originally think. Below are the subtleties in the instantiations.

In the proof of Theorem 2 in [4], after receiving the challenge R from its challenger, the adversary \mathcal{B} , who wishes to solve the discrete log problem, runs the simulator to prove the knowledge of the discrete log of R with respect to base h_1 (by rewinding \mathcal{A} who tries to win the *FindKey* game). Note that in the proof, \mathcal{B} and \mathcal{A} play the roles of the simulator and the verifier, respectively. The simulator \mathcal{S} for Schnorr’s protocol does not need to rewind the (honest) verifier in order to provide a transcript indistinguishable from that of a real interaction. It even does not need to interact with the verifier during the simulation at all, since the verifier is assumed to be honest, and \mathcal{S} can select a random challenge on behalf of the verifier. But in the case here, \mathcal{B} has to interact with \mathcal{A} in order to provide an indistinguishable simulation, thus it has to rewind \mathcal{A} to gain some advantage for the simulation. However, there is only one possible way for \mathcal{B} to rewind \mathcal{A} . That is, after receiving the challenge c from \mathcal{A} , \mathcal{B} rewinds \mathcal{A} back to

the initial state at the beginning of the zero-knowledge proof, and sends \mathcal{A} a new first-round message a , which is computed based on the challenge c obtained from \mathcal{A} . Now, if \mathcal{A} sends back the same c , then the simulation can be completed successfully. However, there is no guarantee on that \mathcal{A} would do so, since \mathcal{A} 's status is changed. In a consequence, \mathcal{B} cannot use \mathcal{S} to provide a zero-knowledge proof desired by \mathcal{A} without the knowledge of $\log_{h_1} R$. Hence, \mathcal{A} may not win the *FindKey* game with probability ϵ again.

An oversimplified way to fix the problem is to let verifier V commit itself to its challenge first before prover P sends its first message. P sends its final response only if V reveals the commitment correctly. It is easy to see that the resulting protocol is zero-knowledge against arbitrary verifier, however, it does not seem to be a proof of knowledge.

A better instantiation of the ZK-POK is to use the efficient 6-round ZK-POK of [8] (which can further be compressed into 4 rounds). The verifier V commits to its challenge and proves to P that he knows the challenge. After that P proves to V that he knows either the challenge or the discrete log. Readers can refer to [8] for details.

We emphasize here again that the above comments do not imply that Goyal's scheme is problematic. Instead, they are regarding to efficient instantiations of the ZK-POK sub-protocol.

3.3 Comments on the Definition of [4]

We also notice that the definition for Traceable IBE given by Goyal in [4] is incomplete and imprecise. Comparing with conventional IBE definition [2], a Traceable IBE scheme described in [4] additionally requires the user to do a “sanity check” on the “well-formedness” of an extracted user secret key from the PKG. However, the meaning of “sanity check” of a user secret key in association with that of “well-formedness” have never been formalized.

Since the PKG is no longer trusted fully in the setting of this research work, the user secret key generated by the PKG using Extract Protocol may be malformed. In this scenario, it is possible that this malformed key can still decrypt a portion of all possible ciphertexts for the user but not all. Now if the malicious PKG publishes another user secret key which can decrypt all the ciphertexts for the user, the user will not be able to win a court if the user provides these two keys as evidence to a court of law, claiming that the PKG is cheating. This is because the PKG can show that one of the keys presented by the user is not a valid key since it cannot decrypt all possible ciphertexts.

Therefore, we believe that the notion of “sanity check” has to be formalized. In addition, in the subsequent security model, we also need to formalize the intuition that a user should be provided with a method to make sure that the user secret key extracted from the PKG via Extract Protocol can always be able to decrypt ciphertexts for the user.

Also due to the lack of “sanity check” definition in [4], the security model of [4] is also incomplete. The attack scenario described above is not captured in any of the models specified in [4].

4 Traceable and Retrievable IBE

4.1 TR-IBE Model

A Traceable and Retrievable Identity-Based Encryption (TR-IBE) scheme consists of six probabilistic polynomial-time (PPT) algorithms and one two-party interactive protocol.

Setup. On input 1^k where $k \in \mathbb{N}$ is a security parameter, it outputs a master public/secret key pair (mpk, msk) .

Extract Protocol. The Private Key Generator, PKG, on input (mpk, msk, ID) carries out the protocol with a user on input (mpk, ID) . At the end of the protocol, the user outputs a user secret key usk_{ID} or a symbol \perp indicating the failure of the protocol run. Formally, the PKG and the user in the protocol are considered as PPT Turing machines. Without loss of generality, we let $ID \in \{0, 1\}^k$. In practice, the user with identity ID may send the identity to the PKG in the first message flow of the protocol.

SanityCheck. On input (mpk, ID, usk_{ID}) , it outputs 1 or 0.

Enc. On input (mpk, ID, m) , where m is a message from a message space \mathcal{M} defined by mpk , it outputs a ciphertext C .

Dec. On input (mpk, usk_{ID}, C) , it outputs $m \in \mathcal{M}$ or a symbol \perp if the decryption fails.

Trace¹. On input (mpk, ID, usk_{ID}) , it outputs \perp if $\text{SanityCheck}(mpk, ID, usk_{ID}) \neq 1$. Otherwise it outputs a user key family number fn_{ID} from a user key family number space denoted by \mathcal{F}_{ID} . This space is defined by (mpk, ID) .

Retrieve. On input $(mpk, ID, usk_{ID}, \widetilde{usk}_{ID})$, it outputs the master secret key msk or a symbol \perp indicating the failure of retrieval.

For correctness, we require that for all $k \in \mathbb{N}$, for any $(mpk, msk) \leftarrow \text{Setup}(1^k)$, any identity $ID \in \{0, 1\}^k$, any $usk_{ID} \neq \perp$ output by the user with identity ID at the end of a run of Extract Protocol with the PKG, any $m \in \mathcal{M}(mpk)$, we have

1. $1 \leftarrow \text{SanityCheck}(mpk, ID, usk_{ID})$;
2. $m \leftarrow \text{Dec}(mpk, usk_{ID}, \text{Enc}(mpk, ID, m))$; and
3. $\text{Trace}(mpk, ID, usk_{ID}) \in \mathcal{F}_{ID}$.

4.2 Security Model for TR-IBE

In [4], three games have been given for formalizing the following three security notions.

1. Confidentiality of Ciphertexts: a conventional indistinguishability based game capturing chosen ciphertext and identity attacks, namely IND-ID-CCA similar to that in [2] is given.

¹ Trace is needed for some technical reason. It is used for formalizing the user key family number and will mainly be used in the Retrieval Game described on page 103.

2. **Secrecy of User Secret Key Against Malicious PKG:** the PKG, after carrying out a successful run of **Extract Protocol** with a user, should not be able to find out the user key family number of the user secret key obtained by the user.
3. **Security Against Framing by Malicious Users:** a user should not be able to come up with two user secret keys such that the corresponding user key family numbers are different, after at most one execution of **Extract Protocol** with the PKG.

As explained in Sec. 3.3, an additional security notion related to “sanity check” of user secret key should be introduced. We also introduce the security notion “Retrievability” which is specific to the retrievability property of TR-IBE. It can be seen that the security against framing by malicious user in TR-IBE is implied by the requirement of confidentiality of ciphertexts and the retrievability property. If the adversary is able to come up with two user secret keys, he can compute the master secret key of PKG due to the retrievability property and is able to break the confidentiality of ciphertexts of all users. In the following, we formalize all these security notions. The corresponding games are Confidentiality (IND-ID-CCA) Game, FindKey Game, SanityCheck Game and Retrievability Game.

Confidentiality (IND-ID-CCA) Game. On input a security parameter 1^k , $k \in \mathbb{N}$, the following game is carried out by a simulator \mathcal{S} against an adversary \mathcal{A} .

1. \mathcal{S} generates $(mpk, msk) \leftarrow \text{Setup}(1^k)$ and invokes \mathcal{A} on mpk . \mathcal{S} maintains a list L . Initially, L is set to empty.
2. \mathcal{A} (which acts as a user) may execute **Extract Protocol** with \mathcal{S} (which acts as the PKG) on any identity ID or query a decryption oracle ODec on (ID, C) . For each run of **Extract Protocol**, if $ID \in L$, \mathcal{S} rejects the protocol run immediately². Otherwise, the protocol is carried out. At the end of a successful run, \mathcal{S} sets $L := L \cup \{ID\}$. For an ODec query, \mathcal{S} generates a user secret key by simulating **Extract Protocol** and uses it to decrypt the querying ciphertext.
3. \mathcal{A} submits two equal length messages $m_0^*, m_1^* \in \mathcal{M}(mpk)$ and an identity ID^* to \mathcal{S} . If $ID^* \in L$, \mathcal{S} aborts. Otherwise, \mathcal{S} flips a random coin $b \xleftarrow{R} \{0, 1\}$ and sends $C^* \leftarrow \text{Enc}(mpk, ID^*, m_b^*)$ to \mathcal{A} . \mathcal{S} sets $L := L \cup \{ID^*\}$.
4. \mathcal{A} can continue executing **Extract Protocol** and querying ODec . At the end of the game, \mathcal{A} outputs its guess b' of b .

\mathcal{A} wins if $b' = b$ and (ID^*, C^*) has never been queried to ODec . The advantage of \mathcal{A} in this game is defined as $\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$.

In the following, we give the formal definition of “Secrecy of User Secret Key Against Malicious PKG”. In Sec. 4.3, we will see that it is stronger than the one given in [4].

FindKey Game. On input a security parameter 1^k , $k \in \mathbb{N}$, the following game is carried out by a simulator \mathcal{S} against an adversary \mathcal{A} .

² This restriction is natural as if \mathcal{A} were allowed to run **Extract Protocol** on an ID for multiple times, due to the retrievability property of TR-IBE, msk may be compromised.

1. \mathcal{S} maintains two lists L_1 and L_2 , both of them are initialized to null. \mathcal{S} invokes \mathcal{A} on 1^k and gets mpk from \mathcal{A} .
2. \mathcal{A} (acts as the malicious PKG) may execute Extract Protocol with \mathcal{S} (which acts as a user) on any identity ID chosen by \mathcal{A} . If $ID \in P(L_1) \cup L_2$, \mathcal{S} rejects the protocol run immediately, where $P(L_1)$ is the collection of the first elements of all the pairs in L_1 . At the end of a successful run, suppose the user secret key generated is usk_{ID} . If $\text{SanityCheck}(mpk, ID, usk_{ID}) = 1$, \mathcal{S} sets $L_1 := L_1 \cup (ID, usk_{ID})$.
3. Since \mathcal{A} (which acts as the malicious PKG) may collude with some users in this multi-user setting (see Sec. 4.3 for more details), \mathcal{A} is allowed to access an oracle called OCorrupt . On input ID , if $(ID, usk_{ID}) \in L_1$ for some user secret key usk_{ID} , the oracle returns usk_{ID} , and sets $L_1 := L_1 \setminus \{(ID, usk_{ID})\}$ and $L_2 := L_2 \cup \{ID\}$. Otherwise, \perp is returned.
4. At the end of the game, \mathcal{A} outputs an identity ID^* and a user secret key \widetilde{usk}_{ID^*} .

\mathcal{A} wins if

1. $1 \leftarrow \text{SanityCheck}(mpk, ID^*, \widetilde{usk}_{ID^*})$;
2. $(ID^*, usk_{ID^*}) \in L_1$; and³
3. $\text{Trace}(mpk, ID^*, usk_{ID^*}) = \text{Trace}(mpk, ID^*, \widetilde{usk}_{ID^*})$.

The advantage of \mathcal{A} in this game is defined as $\Pr[\mathcal{A} \text{ wins}]$.

We now formalize the notion related to “sanity check”. As discussed in Sec. 3.3, a user should be provided with a method to make sure that the user secret key extracted from the PKG via Extract Protocol can always be able to decrypt ciphertexts for the user. We consider the following game. Informally, it requires that for any two user secret keys of an identity that passed the “sanity check”, both of them will always produce the identical result in decryption.

SanityCheck Game. On input a security parameter 1^k , $k \in \mathbb{N}$, a simulator \mathcal{S} invokes an adversary \mathcal{A} on 1^k . \mathcal{A} returns a master public key mpk , an identity ID^* , two user secret keys usk_{ID^*} and \widetilde{usk}_{ID^*} and a ciphertext C . \mathcal{A} wins if

1. $1 \leftarrow \text{SanityCheck}(mpk, ID^*, usk_{ID^*})$;
2. $1 \leftarrow \text{SanityCheck}(mpk, ID^*, \widetilde{usk}_{ID^*})$;
3. $\perp \neq \text{Dec}(mpk, usk_{ID^*}, C)$;
4. $\perp \neq \text{Dec}(mpk, \widetilde{usk}_{ID^*}, C)$; and
5. $\text{Dec}(mpk, usk_{ID^*}, C) \neq \text{Dec}(mpk, \widetilde{usk}_{ID^*}, C)$.

The advantage of \mathcal{A} in this game is defined as $\Pr[\mathcal{A} \text{ wins}]$. By combining the notion captured in this game and the correctness requirement defined at the beginning of Sec. 4, it is easy to see that the intuition of “sanity check” is captured.

³ Note that we do not need the restriction that $ID^* \notin L_2$ as ID^* cannot co-exist in both L_1 and L_2 .

The last security notion, also the only one specific to “Retrievability” property, requires that for any two user secret keys corresponding to the same identity but with different user key family numbers, they allow the public to retrieve the master secret key. The following game considers a malicious PKG which tries to come up with two user secret keys such that its master secret key is not retrievable.

Retrievability Game. On input a security parameter 1^k , $k \in \mathbb{N}$, a simulator \mathcal{S} invokes an adversary \mathcal{A} on 1^k . \mathcal{A} returns a master public key mpk , an identity ID^* , and two user secret keys usk_{ID^*} and \widetilde{usk}_{ID^*} . \mathcal{A} wins if

1. $1 \leftarrow \text{SanityCheck}(mpk, ID^*, usk_{ID^*})$;
2. $1 \leftarrow \text{SanityCheck}(mpk, ID^*, \widetilde{usk}_{ID^*})$;
3. $\text{Trace}(mpk, ID^*, usk_{ID^*}) \neq \text{Trace}(mpk, ID^*, \widetilde{usk}_{ID^*})$; and
4. $\perp \leftarrow \text{Retrieve}(mpk, ID^*, usk_{ID^*}, \widetilde{usk}_{ID^*})$.

The advantage of \mathcal{A} in this game is defined as $\Pr[\mathcal{A} \text{ wins}]$.

Theorem 1. *A TR-IBE scheme is said to be secure if for all polynomial time adversaries, the advantage in each of the Confidentiality game, FindKey game, SanityCheck game and Retrievability game is negligible in the security parameter k .*

4.3 Further Comments on the Security Model of [4]

The FindKey Game defined in [4] is weaker than our model defined above. In particular, the game in [4] requires the adversary to fix the challenging identity ID^* at the beginning of the game and no further change is allowed. Also, the adversary is not allowed to interact with other identities. In our definition instead, we allow the adversary to “try out” and also corrupt a couple of identities in the manner of *adaptive chosen identity attack* before choosing a challenging identity at the end of the game.

5 A Concrete Scheme

5.1 Construction

High Level Description. Our scheme is based on Gentry’s IBE scheme [3]. We extend the scheme by adding a Verifiable Encryption (VE) scheme [9]. The PKG has two sets of public key pair. One is for the IBE and the other for the VE. At the beginning, the PKG verifiably encrypts, using its VE public key, the (IBE) master secret key. The encrypted master secret key is published. In the key extraction process, one additional component is given to the user as the secret key. This component allows the revocation of the encrypted master secret key. That is, if the PKG generates two secret keys (they may be different) for the same user, by using these two secret keys, the master secret key can be decrypted and revoked.

Technical Details.

Setup: On input 1^k , where $k \in \mathbb{N}$ is a security parameter, let \mathbb{G} and \mathbb{G}_T be groups of order p such that p is a k -bit prime, and let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be the bilinear map. We use a multiplicative notation for the operation in \mathbb{G} and \mathbb{G}_T . Let $H_I : \{0, 1\}^* \rightarrow \mathbb{Z}_p$, $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$, $H_t : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be secure cryptographic hash functions. The PKG selects four random generators $g, h_1, h_2, h_3 \in \mathbb{G}$ and randomly chooses $\alpha, \beta, x \in_R \mathbb{Z}_p$. It sets $g_1 = g^\alpha, g' = g^\beta$ and $X = e(g, g)^x$. Define the message space $\mathcal{M} = \mathbb{G}_T$.

Then the PKG runs the non-interactive verifiable encryption algorithm from [9]. The idea is to verifiably encrypt the master secret key α and β under the public key X . This can be done as follows. Let $N := \text{poly}(k)$ for some polynomial $\text{poly}(\cdot)$, be a security parameter⁴. Let $H_E : \{0, 1\}^* \rightarrow \{0, 1\}^N$ be a secure hash function.

1. The PKG randomly selects $u_j, u'_j \in_R \mathbb{Z}_p$ and computes $(T_j = g^{u_j}, T'_j = g^{u'_j})$, for $j = 1$ to N .
2. For $j = 1$ to N the PKG computes the following.
 - Compute $Z_{j,0} = u_j, Z_{j,1} = u_j - \alpha, Z'_{j,0} = u'_j, Z'_{j,1} = u'_j - \beta$.
 - Randomly select $v_{j,0}, v_{j,1}, v'_{j,0}, v'_{j,1} \in_R \mathbb{Z}_p$, compute $E_{0,j,i} = X^{v_{j,i}} \oplus Z_{j,i}, E_{1,j,i} = g^{v_{j,i}}$ and $E'_{0,j,i} = X^{v'_{j,i}} \oplus Z'_{j,i}, E'_{1,j,i} = g^{v'_{j,i}}$ for $i \in \{0, 1\}$ ⁵.
3. PKG computes $L = H_E(T_1 || E_{0,1,0} || E_{1,1,0} || E_{0,1,1} || E_{1,1,1} || \dots || T_N || E_{0,N,0} || E_{1,N,0} || E_{0,N,1} || E_{1,N,1})$ and $L' = H_E(T'_1 || E'_{0,1,0} || E'_{1,1,0} || E'_{0,1,1} || E'_{1,1,1} || \dots || T'_N || E'_{0,N,0} || E'_{1,N,0} || E'_{0,N,1} || E'_{1,N,1})$. Let b_j, b'_j be the j -th bit of L and L' respectively.
4. Output $\mathcal{T} = \{ (T_j, T'_j, E_{0,j,0}, E_{1,j,0}, E_{0,j,1}, E_{1,j,1}, E'_{0,j,0}, E'_{1,j,0}, E'_{0,j,1}, E'_{1,j,1}, Z_{j,b_j}, v_{j,b_j}, Z'_{j,b'_j}, v'_{j,b'_j}) \}_{j=1}^N$.
5. Verification. Anyone can check if \mathcal{T} is a valid encryption of α and β under the public key X by computing L and L' from \mathcal{T} and checking if the following equations hold:

$$\begin{aligned}
 E_{0,j,b_j} &\stackrel{?}{=} X^{v_{j,b_j}} \oplus Z_{j,b_j} \\
 E_{1,j,b_j} &\stackrel{?}{=} g^{v_{j,b_j}} \\
 T_j &\stackrel{?}{=} g_1^{b_j} g^{Z_{j,b_j}} \\
 E'_{0,j,b'_j} &\stackrel{?}{=} X^{v'_{j,b'_j}} \oplus Z'_{j,b'_j} \\
 E'_{1,j,b'_j} &\stackrel{?}{=} g^{v'_{j,b'_j}} \\
 T'_j &\stackrel{?}{=} g^{b'_j} g^{Z'_{j,b'_j}}
 \end{aligned}$$

where $j = 1$ to N and b_j, b'_j are the j -th bit of L and L' respectively.

The public parameters *mpk* and master secret key *msk* are given by

$$mpk = (g, g_1, g', h_1, h_2, h_3, X, H, H_I, H_t, \mathcal{M}, \mathcal{T}) \quad msk = (\alpha, \beta, g^x)$$

⁴ N controls the cheating probability of the verifiable encryption.

⁵ We assume that some appropriate padding has been added for the \oplus operation.

Extract Protocol: On input the master public key/secret key pair (mpk, msk) and an identity $ID \in \{0, 1\}^k$ of a user U , the PKG carries out an interactive protocol with U , as follows. Compute $\mathbb{ID} = H_I(ID)$. If $\mathbb{ID} = \alpha$, it aborts. Otherwise, the protocol proceeds as follow:

1. The user U selects a random $r \in_R \mathbb{Z}_p$ and sends $R = h_1^r$ to the PKG.
2. U gives to the PKG the following zero-knowledge proof of knowledge:

$$PK\{r : R = h_1^r\}$$

3. The PKG randomly selects $r', r_{\mathbb{ID},2}, r_{\mathbb{ID},3} \in \mathbb{Z}_p$ and computes

$$\begin{aligned} h'_{\mathbb{ID}} &= (Rg^{-r'})^{1/(\alpha-\mathbb{ID})} & t_{\mathbb{ID}} &= g^x h_1^{\frac{H_t(R,r')}{\beta+\mathbb{ID}}} \\ h_{\mathbb{ID},2} &= (h_2g^{-r_{\mathbb{ID},2}})^{1/(\alpha-\mathbb{ID})} & h_{\mathbb{ID},3} &= (h_3g^{-r_{\mathbb{ID},3}})^{1/(\alpha-\mathbb{ID})} \end{aligned}$$

and sends $(r', h'_{\mathbb{ID}}, t_{\mathbb{ID}}, r_{\mathbb{ID},2}, h_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},3})$ to U .

4. The PKG computes $\pi_{\mathbb{ID}}$, the non-interactive proof statement of the following zero-knowledge proof of knowledge:

$$\begin{aligned} \pi_{\mathbb{ID}} = PK\left\{ (x, \alpha, \beta) : X = e(g, g)^x \wedge g_1 = g^\alpha \wedge g' = g^\beta \wedge \right. \\ \left. h'_{\mathbb{ID}} = R^{\frac{1}{\alpha-\mathbb{ID}}} g^{\frac{-r'}{\alpha-\mathbb{ID}}} \wedge t_{\mathbb{ID}} = g^x h_1^{\frac{H_t(R,r')}{\beta+\mathbb{ID}}} \right\} \end{aligned}$$

and sends to U .

5. After checking $\pi_{\mathbb{ID}}$, U computes

$$r_{\mathbb{ID},1} = r'/r \quad h_{\mathbb{ID},1} = (h'_{\mathbb{ID}})^{1/r}$$

The secret key $usk_{\mathbb{ID}}$ is $(r, r_{\mathbb{ID},1}, h_{\mathbb{ID},1}, r_{\mathbb{ID},2}, h_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}})$ where

$$h_{\mathbb{ID},1} = (h_1g^{-r_{\mathbb{ID},1}})^{1/(\alpha-\mathbb{ID})} \quad \text{and} \quad t_{\mathbb{ID}} = g^x h_1^{\frac{H_t(h_1^r, r_{\mathbb{ID},1})}{\beta+\mathbb{ID}}}.$$

SanityCheck: On input $(mpk, usk_{\mathbb{ID}})$ and an identity $ID \in \{0, 1\}^k$, compute $\mathbb{ID} = H_I(ID)$ and check whether $e(h_{\mathbb{ID},i}, g_1/g^{\mathbb{ID}}) = e(h_i g^{-r_{\mathbb{ID},i}}, g)$ for $i = 1, 2, 3$. Also check whether $\pi_{\mathbb{ID}}$ is a valid proof statement. If all of them are correct, output 1. Otherwise, output 0.

Enc: To encrypt a message $m \in \mathbb{G}_T$ using identity $ID \in \{0, 1\}^k$, compute $\mathbb{ID} = H_I(ID)$, generate a random $s \in_R \mathbb{Z}_p$ and output the ciphertext C where:

$$\begin{aligned} C &= (C_1, C_2, C_3, C_4, C_5) \\ &= \left(g_1^s g^{-s\mathbb{ID}}, e(g, g)^s, m \cdot e(g, h_1)^{-s}, e(g, h_2)^s e(g, h_3)^{s\gamma}, \pi_C \right) \end{aligned}$$

where $\gamma = H(C_1, C_2, C_3)$ and π_C is a non-interactive proof statement of the following zero-knowledge proof of knowledge

$$\pi_C = PK\left\{ (s) : C_1 = (g_1 g^{-\mathbb{ID}})^s \wedge C_2 = e(g, g)^s \right\}$$

Dec: To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$ using secret key usk_{ID} , compute $\gamma = H(C_1, C_2, C_3)$ and test whether

$$C_4 = e(C_1, h_{\text{ID},2} h_{\text{ID},3}^\gamma) \cdot C_2^{r_{\text{ID},2} + r_{\text{ID},3} \gamma}$$

If it is not equal or π_C is not a valid proof statement, output \perp . Else output

$$m = C_3 \cdot e(C_1, h_{\text{ID},1}) \cdot C_2^{r_{\text{ID},1}}$$

Trace: On input $(mpk, \text{ID}, usk_{\text{ID}})$, define family space $\mathcal{F}_{\text{ID}} = \mathbb{Z}_p$. Parse $usk_{\text{ID}} = (r, r_{\text{ID},1}, h_{\text{ID},1}, r_{\text{ID},2}, h_{\text{ID},2}, r_{\text{ID},3}, h_{\text{ID},3}, t_{\text{ID}}, \pi_{\text{ID}})$ and output the decryption key family number $f_{n_{\text{ID}}} = r_{\text{ID},1}$.

Retrieve: On input (mpk, ID) and two sets of secret key $(usk_{\text{ID}}, \widetilde{usk}_{\text{ID}})$ for the same identity ID ,

1. Compute $\mathbb{ID} = H_I(\text{ID})$ and parse $usk_{\text{ID}} = (r, r_{\text{ID},1}, h_{\text{ID},1}, r_{\text{ID},2}, h_{\text{ID},2}, r_{\text{ID},3}, h_{\text{ID},3}, t_{\text{ID}}, \pi_{\text{ID}})$ and $\widetilde{usk}_{\text{ID}} = (\tilde{r}, \tilde{r}_{\text{ID},1}, \tilde{h}_{\text{ID},1}, \tilde{r}_{\text{ID},2}, \tilde{h}_{\text{ID},2}, \tilde{r}_{\text{ID},3}, \tilde{h}_{\text{ID},3}, \tilde{t}_{\text{ID}}, \tilde{\pi}_{\text{ID}})$
2. Compute $K := H_t(h_1^r, rr_{\text{ID},1})$ and $\tilde{K} := H_t(h_1^{\tilde{r}}, \tilde{r}\tilde{r}_{\text{ID},1})$. If $K = \tilde{K}$, output \perp . Otherwise, compute

$$\left(\frac{t_{\text{ID}}^{\tilde{K}}}{\tilde{t}_{\text{ID}}^K} \right)^{\frac{1}{\tilde{K} - K}} = g^x \tag{1}$$

and check whether $X \stackrel{?}{=} e(g, g^x)$. If not, output \perp .

3. For any $j \in \{1, \dots, N\}$, one can get $(T_j, Z_{j,b_j}, E_{0,j,1-b_j}, E_{1,j,1-b_j})$ in \mathcal{T} (the verifiable encryption in Setup). For simplicity, we omit the subscript j . That is, one can get $(T := g^u, Z_b, E_{0,1-b}, E_{1,1-b} := g^{v_{1-b}})$, such that $b \in \{0, 1\}$ where

$$Z_b = u - b\alpha \tag{2}$$

Compute $e(E_{1,1-b}, g^x) \oplus E_{0,1-b}$ to get

$$Z_{1-b} = u - (1-b)\alpha \tag{3}$$

From equation (2) and (3), compute α . Check whether $g_1 \stackrel{?}{=} g^\alpha$. If not, use another $j \in \{1, \dots, N\}$ to compute α . If all j 's have been used and no equality is attained, output \perp . Otherwise, compute β in the similar way and output (α, β, g^x) as msk .

5.2 Security Analysis

Theorem 2. *The advantage of an adversary in the IND-ID-CCA game is negligible for the proposed scheme under the decisional truncated q -ABDHE assumption in the random oracle model.*

Proof: (sketch) The proof in our setting very much falls along the lines of the proof of IND-ID-CCA security of Goyal's scheme [4]. Here we just give a sketch highlighting the differences.

The differences between [4] and our scheme (in terms of IND-ID-CCA) are the formation of public parameter and the generation of a decryption key. In our scheme, $mpk = (g, g_1, g', h_1, h_2, h_3, X, H, H_t, H_I, \mathcal{M}, \mathcal{T})$ where $(g, g_1, h_1, h_2, h_3, \mathcal{M})$ are generated in the same way as in [4]. The remaining parameters are generated as follows. The simulator \mathcal{S} randomly generates $x, \beta \in_R \mathbb{Z}_p$ and sets $g' = g^\beta, X = e(g, g)^x$. \mathcal{S} also moderates the hash functions H, H_I, H_t as random oracles. By controlling the random oracles, \mathcal{S} can easily simulate the transcript \mathcal{T} .

In the Extraction Oracle, we use the same technique as in [4] to extract r from the user and set $r' = rr_{\mathbb{ID},1}$. \mathcal{S} can generate $t_{\mathbb{ID}}$ easily, as it knows x, β .

\mathcal{S} can also simulate the transcript $x_{\mathbb{ID}}$ by controlling the random oracle. \square

Theorem 3. *The advantage of an adversary in the FindKey game is negligible for the proposed scheme if the discrete log problem in \mathbb{G} is hard.*

Proof: The proof follows along the lines of the proof for the FindKey game in [4], except that we are in the adaptive chosen identity attack model.

Let \mathcal{A} denote a PPT algorithm that has a non-negligible advantage ϵ in winning the FindKey game, we construct another PPT algorithm \mathcal{B} that breaks the discrete log assumption in \mathbb{G} with a non-negligible probability. \mathcal{B} proceeds as follows.

\mathcal{B} runs the algorithm \mathcal{A} and gets the public parameters $mpk = (g, g_1, g', h_1, h_2, h_3, X, H, H_I, H_t, \mathcal{M}, \mathcal{T})$. \mathcal{B} pass h_1 to its challenger and gets a challenge $R \in \mathbb{G}$. \mathcal{B} 's goal is to find $r \stackrel{def}{=} \log_{h_1} R$.

Assume adversary \mathcal{A} makes at most q_I extract queries, after getting a challenge R from the challenger, \mathcal{B} selects $i \stackrel{R}{\leftarrow} \{1, 2, 3, \dots, q_I\}$. For each $1 \leq j \leq q_I$, if $j = i$, \mathcal{B} sets $R_j = R$, otherwise, \mathcal{B} randomly selects $r_j \in \mathbb{Z}_p$ and sets $R_j = h_1^{r_j}$.

\mathcal{B} answers \mathcal{A} 's queries as follows:

- Extract queries. When \mathcal{A} performs an extract query on an identity ID_j , \mathcal{B} rejects the query if ID_j has already been created. Otherwise, \mathcal{B} performs the extract protocol as follows: if $j \neq i$, \mathcal{B} runs the extract protocol as usual; if $j = i$, \mathcal{B} gives R to \mathcal{A} together with a zero-knowledge proof. The zero-knowledge proof is generated by rewinding \mathcal{A} . Note that here we require \mathcal{A} to commit to the challenge before the zero-knowledge proof is carried out so that when rewinding \mathcal{A} the same challenge will be used by \mathcal{A} . In this way, \mathcal{B} is able to simulate a proof by running the simulator of the zero knowledge proof system without the knowledge of r . After receiving $(r', h'_{\mathbb{ID}_i}, t'_{\mathbb{ID}_i}, r_{\mathbb{ID}_i,2}, h_{\mathbb{ID}_i,2}, r_{\mathbb{ID}_i,3}, h_{\mathbb{ID}_i,3})$ and $\pi_{\mathbb{ID}_i}$ from \mathcal{A} , \mathcal{B} verifies $\pi_{\mathbb{ID}_i}$ and runs a key sanity check by testing if $e(h_{\mathbb{ID}_i,t}, g_1/g^{\mathbb{ID}_i}) = e(h_t g^{-r_{\mathbb{ID}_i,t}}, g)$ for $t = 2, 3$. For $t = 1$, \mathcal{B} tests if $e(h'_{\mathbb{ID}_i}, g_1/g^{\mathbb{ID}_i}) = e(R_i g^{-r'}, g)$. If any of these tests fails, \mathcal{B} aborts with failure, otherwise, ID_i is added to L_1 , notice that \mathcal{B} cannot derive the final user secret key usk_{ID_i} in this case.

- Corruption queries. When \mathcal{A} performs a corruption query on identity $\hat{\text{ID}}$, if $\hat{\text{ID}} \in L_1$:
 - (a) $\hat{\text{ID}} = \text{ID}_i$, \mathcal{B} aborts with failure
 - (b) $\hat{\text{ID}} \neq \text{ID}_i$, \mathcal{B} returns $usk_{\hat{\text{ID}}}$ to \mathcal{A} otherwise, \mathcal{B} rejects the query.

Finally, if \mathcal{B} does not abort the game, with probability ϵ , \mathcal{A} will output a decryption key (passing the key sanity check) usk'_{ID_n} which has the same family number with usk_{ID_n} in L_1 . If $n = i$, then \mathcal{B} can calculate $r = r' / f'_{n_{\text{ID}_i}}$ where $f'_{n_{\text{ID}_i}}$ is the key family number of usk'_{ID_i} . It is obvious that $n = i$ implies that usk_{ID_i} is in L_1 and \mathcal{B} does not abort in the game. Since i is randomly chosen, \mathcal{B} 's success probability in solving the discrete log problem in \mathbb{G} is at least $\frac{\epsilon}{q}$. \square

Theorem 4. *The advantage of an adversary in the SanityCheck game is negligible for the proposed scheme.*

Proof: (sketch) Let the output of \mathcal{A} be $mpk, \text{ID}^*, C = (C_1, C_2, C_3, C_4, C_5)$, $usk_{\text{ID}^*} = \{r, r_{\text{ID},1}, r_{\text{ID},2}, r_{\text{ID},3}, h_{\text{ID},1}, h_{\text{ID},2}, h_{\text{ID},3}, t_{\text{ID}}, \pi_{\text{ID}}\}$ and $usk_{\text{ID}^*} = \{\tilde{r}, \tilde{r}_{\text{ID},1}, \tilde{r}_{\text{ID},2}, \tilde{r}_{\text{ID},3}, \tilde{h}_{\text{ID},1}, \tilde{h}_{\text{ID},2}, \tilde{h}_{\text{ID},3}, \tilde{t}_{\text{ID}}, \tilde{\pi}_{\text{ID}}\}$. \mathcal{A} wins implies that condition (1) - (5) defined in section 4.2 are all fulfilled.

Condition (1) implies

$$\begin{aligned} e(h_{\text{ID},i}, g_1/g^{\text{ID}}) &= e(h_i g^{-r_{\text{ID},i}}, g) \\ \Rightarrow e(h_{\text{ID},i}, g^{\alpha-\text{ID}}) &= e(h_i g^{-r_{\text{ID},i}}, g) \\ \Rightarrow e((h_{\text{ID},i})^{\alpha-\text{ID}}, g) &= e(h_i g^{-r_{\text{ID},i}}, g) \\ \Rightarrow (h_{\text{ID},i})^{\alpha-\text{ID}} &= (h_i g^{-r_{\text{ID},i}}) \\ \Rightarrow h_{\text{ID},i} &= (h_i g^{-r_{\text{ID},i}})^{\frac{1}{\alpha-\text{ID}}} \quad \text{for } i = 1, 2, 3 \end{aligned} \tag{4}$$

Similarly, condition (2) implies

$$\tilde{h}_{\text{ID},i} = (h_i g^{-\tilde{r}_{\text{ID},i}})^{\frac{1}{\alpha-\text{ID}}} \quad \text{for } i = 1, 2, 3 \tag{5}$$

Condition (3) and (4) imply that C_5 verifies (that is, C_5 is a valid SPK). That is, in the random oracle model, the simulator can extract s such that

$$C_1 = (g^{\alpha-\text{ID}})^s \quad \text{and} \quad C_2 = e(g, g)^s \tag{6}$$

Condition (5) implies that

$$C_3 \cdot e(C_1, h_{\text{ID},1}) \cdot C_2^{r_{\text{ID},1}} \neq C_3 \cdot e(C_1, \tilde{h}_{\text{ID},1}) \cdot C_2^{\tilde{r}_{\text{ID},1}} \tag{7}$$

We have

$$\begin{aligned} LHS &= C_3 \cdot e(C_1, h_{\text{ID},1}) \cdot C_2^{r_{\text{ID},1}} \\ &= C_3 \cdot e\left((g^{\alpha-\text{ID}})^s, (h_1 g^{-r_{\text{ID},1}})^{\frac{1}{\alpha-\text{ID}}}\right) \cdot e(g, g)^{s \cdot r_{\text{ID},1}} \text{ from equation (4) and (6)} \\ &= C_3 \cdot e(g^s, h_1 g^{-r_{\text{ID},1}}) \cdot e(g, g)^{s \cdot r_{\text{ID},1}} \\ &= C_3 \cdot e(g, h_1)^s \cdot e(g, g)^{s \cdot (-r_{\text{ID},1})} \cdot e(g, g)^{s \cdot r_{\text{ID},1}} \\ &= C_3 \cdot e(g, h_1)^s \end{aligned}$$

Similarly we have

$$\begin{aligned} RHS &= C_3 \cdot e(C_1, \tilde{h}_{\mathbb{ID},1}) \cdot C_2^{\tilde{r}_{\mathbb{ID},1}} \\ &= C_3 \cdot e(g, h_1)^s = LHS \end{aligned} \quad (8)$$

However, equation (8) contradicts to equation (7). Thus \mathcal{A} wins the game only with negligible probability. \square

Theorem 5. *The advantage of an adversary in the Retrieval game is negligible for the proposed scheme.*

Proof: (sketch) Let the output of \mathcal{A} be $mpk, \mathbb{ID}^*, C = (C_1, C_2, C_3, C_4, C_5)$, $usk_{\mathbb{ID}^*} = \{r, r_{\mathbb{ID},1}, r_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},1}, h_{\mathbb{ID},2}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}}\}$ and $\widetilde{usk}_{\mathbb{ID}^*} = \{\tilde{r}, \tilde{r}_{\mathbb{ID},1}, \tilde{r}_{\mathbb{ID},2}, \tilde{r}_{\mathbb{ID},3}, \tilde{h}_{\mathbb{ID},1}, \tilde{h}_{\mathbb{ID},2}, \tilde{h}_{\mathbb{ID},3}, \tilde{t}_{\mathbb{ID}}, \tilde{\pi}_{\mathbb{ID}}\}$ such that $r_{\mathbb{ID},1} \neq \tilde{r}_{\mathbb{ID},1}$. \mathcal{A} wins implies that condition (1) - (4) defined in section 4.2 are all fulfilled.

Condition (1) and (2) implies that the PK on $\pi_{\mathbb{ID}}$ and $\tilde{\pi}_{\mathbb{ID}}$ are sound. That is, $\left(\frac{t_{\mathbb{ID}}^{\tilde{K}}}{\tilde{t}_{\mathbb{ID}}}\right)^{\frac{1}{\tilde{K}-K}} = g^x$. Condition (3) implies that $r_{\mathbb{ID},1} \neq \tilde{r}_{\mathbb{ID},1}$, that is, $K := H_t(g^r, r_{\mathbb{ID},1}r) \neq \tilde{K} := H_t(g^{\tilde{r}}, \tilde{r}_{\mathbb{ID},1}\tilde{r})$. Condition (4) implies that either

1. $K \neq \tilde{K}$; or
2. $X \neq e(g, g^x)$ where g^x is computed from equation (1); or
3. $g_1 \neq g^\alpha$ where α is computed from equation (2) and (3)

Case (1) happens with negligible probability, due to the collision resistance property of the hash function H_t . Case (2) happens with negligible probability, due to the soundness of SanityCheck which has been proven above. Case (3) also happens with negligible probability, due to the security of the verifiable encryption scheme [9].

Combining all cases, the adversary only has negligible advantage to win the game. \square

6 Conclusion

In this paper, we firstly identified a security issue of Goyal's work in [4, 5]. We then proposed a way to fix it. Then, we took one step further than Goyal's work by proposing a Traceable and Retrievable IBE. In our notion, the PKG's master secret key is retrieved automatically if more than one user secret key are released. We presented a formal model to capture this idea, and proposed a concrete scheme based on Gentry's IBE [3]. We believe that the model we proposed in this paper may be more appealing in practice as our model encourages the PKG to strengthen their security system. If someone can successfully release an additional user secret key, it means that his security is also compromised.

Acknowledgements

We would like to thank the anonymous reviewers of ACNS 2008 for their suggestions and invaluable comments to improve this paper.

References

1. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
2. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
3. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaude- nay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
4. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–448. Springer, Heidelberg (2007)
5. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. Cryptology ePrint Archive, Report 2007/368 (2007); revised and extended version of [4], <http://eprint.iacr.org/2007/368>
6. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 235–251. Springer, Heidelberg (1990)
7. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report 260, Institute for Theoretical Computer Science, ETH Zurich (1997)
8. Cramer, R., Damgård, I., MacKenzie, P.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
9. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000)

Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures

Takashi Nishide^{1,2}, Kazuki Yoneyama¹, and Kazuo Ohta¹

¹ Department of Information and Communication Engineering,
The University of Electro-Communications, 1-5-1 Chofugaoka Chofu-shi,
Tokyo 182-8585 Japan

{t-nishide,yoneyama,ota}@ice.uec.ac.jp

² Hitachi Software Engineering Co., Ltd.; 4-12-7 Higashi-Shinagawa Shinagawa-ku,
Tokyo, 140-0002 Japan

Abstract. We propose attribute-based encryption schemes where encryptor-specified access structures (also called ciphertext policies) are hidden. By using our schemes, an encryptor can encrypt data with a hidden access structure. A decryptor obtains her secret key associated with her attributes from a trusted authority in advance and if the attributes associated with the decryptor's secret key do not satisfy the access structure associated with the encrypted data, the decryptor cannot decrypt the data or guess even what access structure was specified by the encryptor. We prove security of our construction based on the Decisional Bilinear Diffie-Hellman assumption and the Decision Linear assumption. In our security notion, even the legitimate decryptor cannot obtain the information about the access structure associated with the encrypted data more than the fact that she can decrypt the data.

Keywords: Attribute-Based Encryption, Recipient Anonymity, Access Control on Encrypted Data, Ciphertext Policy.

1 Introduction

In the distributed setting, we need to enforce access control policies to protect various resources. In such settings, it may be suitable to specify access control policies based on attributes rather than individual identities, because an identity may not have enough information about its entity. Attribute-based encryption (ABE) is a mechanism by which we can realize such access control in a cryptographic way. There are two kind of ABE schemes, key-policy and ciphertext-policy ABE schemes.

In the key-policy ABE schemes [12,18,19,15], ciphertexts are associated with sets of attributes and users' secret keys are associated with access structures. If the attributes associated with the ciphertext satisfy the access structure of the secret key, the secret key holder can decrypt the ciphertext successfully. Also the concept of searchable and predicate encryption [5,21] is related to key-policy ABE in the sense that successful decryption is conditional on access structure associated with secret keys.

On the other hand, in the ciphertext-policy ABE (CP-ABE) schemes [2, 11, 15, 16], the situation is reversed. That is, attributes are associated with secret keys and access structures are associated with ciphertexts and called ciphertext policies. The access structures are described with the attributes and therefore the concept of CP-ABE is closely related to Role-Based Access Control.

In this work, we focus on CP-ABE and construct a CP-ABE scheme where we can hide encryptor-specified access structures associated with ciphertexts. Our scheme can be considered as a recipient-anonymous targeted broadcast and the relation of our scheme to a normal CP-ABE scheme is similar to that of anonymous identity-based encryption (IBE) to normal IBE. For example, suppose a company wants to hire certain qualified people who satisfy the policy the company specified and the policy may contain the useful information about the company's business strategy. The company can post a message encrypted by our CP-ABE scheme on a public bulletin board to seek applications. By doing so, the company can keep the important policy confidential. Since the policy is hidden, the rival companies cannot know what kind of policy the company used to hire its employees.

In the ABE schemes, *collusion-resistance* is an important property. We do not want the secret key holders to be able to combine their secret keys to decrypt ciphertexts neither of them can decrypt. By building on the previous schemes [2, 11], we can also realize collusion-resistant CP-ABE schemes.

Our Results. We construct two CP-ABE schemes with partially hidden ciphertext policies in the sense that possible values of each attribute in the system should be known to an encryptor in advance and the encryptor can hide what subset of possible values for each attribute in the ciphertext policy can be used for successful decryption. In our schemes, encryptors can use wildcards to mean that certain attributes are not relevant to the ciphertext policy in a hidden way. The security proof of our first construction is given under the Decisional Bilinear Diffie-Hellman assumption and the Decision Linear assumption. Since these assumptions are general, we can use a large variety of elliptic curves (including both asymmetric and symmetric bilinear pairings) to implement our first scheme though we use the symmetric notation for ease of exposition. The security proof of our second construction is given in the generic group model and the second construction needs DDH-hard groups, but with a property inherited from [2], the second construction is more flexible than the first construction in that new attributes can be added in the ciphertext policy securely with the existing public parameters being unchanged even after the system setup is done. We mention this aspect in Sect. 5 in more detail. We describe our constructions in the multi-valued attribute setting where an attribute can take multiple values and this setting is a generalization of the access structures used in [11]. In our security notion, even the legitimate decryptor cannot obtain the information about the ciphertext policy more than the fact that she can decrypt the data.

Related Work. Bethencourt, Sahai, and Waters [2] proposed the first CP-ABE scheme. Their scheme allows the ciphertext policies to be very expressive, but the security proof is in the generic group model and the policies need to

be revealed in the ciphertexts because decryptors must know how they should combine their secret key components for decryption. Cheung and Newport [11] proposed a provably secure CP-ABE scheme and their scheme deals with negative attributes explicitly and supports wildcards in the ciphertext policies but the policies need to be revealed as in [2]. Kapadia et al. [14] also proposed a CP-ABE scheme and their scheme realizes hidden ciphertext policies that have the same expressiveness as [11], but their scheme is not collusion-resistant and needs an online semi-trusted server that must know the attributes' values every user in the system has and re-encrypt ciphertexts for each user when the user retrieves the ciphertexts. Such an online semi-trusted server can be a performance bottleneck in the system while, in our schemes, encryptors can just post or broadcast ciphertexts. Lubicz and Sirvent [16] proposed another CP-ABE scheme that has the same expressiveness as [11] and only 3 pairing computations are needed for decryption, but the ciphertext policies need to be revealed for decryption. Shi et al. [21] proposed a predicate encryption scheme that focuses on range queries over huge numbers, the dual of which can also realize a CP-ABE scheme where an encryptor can specify a number range in the ciphertext policy. The security proof of [21] is based on the security notion weaker than ours, which is called *match-revealing security* in [21] and the number of attributes must be small because the decryption cost is exponential in the number of attributes. Boneh and Waters [5] proposed a predicate encryption scheme based on the primitive called *Hidden Vector Encryption* or HVE for short. The scheme in [5] can realize the same functionality as ours by using the opposite semantics of subset predicates described in [5] (see Appendix A for the details). However, it needs bilinear groups the order of which is a product of two large primes, so it needs to deal with large group elements and the numbers of both attributes and possible values for each attribute specified in the ciphertext policy are fixed at the system setup while, in our constructions, the number of possible attribute values in the ciphertext policy can be increased and furthermore in our second construction, the number of attributes in the ciphertext policy can be increased securely even after the system setup with the existing public parameters being unchanged.

Recently, Katz, Sahai, and Waters [15] proposed a novel predicate (or *functional*) encryption scheme supporting *inner product* predicates and their scheme is very general and can realize both key-policy and ciphertext-policy ABE schemes. Their scheme can also realize hidden ciphertext policies that can be more expressive than ours. However, their scheme is based on a special type of bilinear groups the order of which is a product of three (or two) large primes while ours are not. Therefore, their scheme needs to deal with large group elements and requires new complexity assumptions for the security proof. By using the dual of the predicate corresponding to polynomial evaluation, the scheme in [15] can realize the same access structure of ciphertext policies that our schemes can support (see Appendix B for the details) and then the ciphertext size of our schemes $O(\sum_{i=1}^n n_i)$ is comparable to that of [15] where n is the number of attributes in ciphertext policies and n_i is the number of possible values for each attribute i . For example, if attribute i is boolean, $n_i = 2$. In the CP-ABE

Table 1. Comparison of different schemes

	Expressiveness of policy	Anonymity	Complexity assumption	Type of bilinear group	Add attrs after setup
[5]	AND-gates on multi-valued attributes with wildcards	yes	cDBDH, C3DH	group of composite order $N = pq$	no
[2]	all boolean formula	no	generic group model	any	yes
[11]	AND-gates on positive and negative attributes with wildcards	no	DBDH	any	no
[15]	all boolean formula	yes	new assumptions based on composite order group	group of composite order $N = pqr$	no
This work1	AND-gates on multi-valued attributes with wildcards	yes	DBDH, D-Linear	any	no
This work2	AND-gates on multi-valued attributes with wildcards	yes	generic group model	DDH-hard group	yes

scheme of [15], the maximum size of the subset of attribute values for each attribute specified in the ciphertext policy for successful decryption is fixed at the system setup while, in our constructions, the size can be increased. Also, the number of attributes specified in the ciphertext policy is fixed at the system setup while, in our second construction, the number of attributes in the ciphertext policy can be increased securely even after the system setup with the existing public parameters being unchanged. However, when the number of possible attribute values is huge, the scheme in [15] is more advantageous than ours because it can enjoy the smaller ciphertext size and still realize the wildcard functionality.

Chase [10] proposed a multi-authority ABE where multiple authorities generate secret keys for their monitored attributes. The technique of [10] can be applicable to our schemes too. Abdalla et al. [1] proposed an identity-based encryption scheme where an encryptor can use wildcards to specify recipients of the ciphertext, but the positions of the wildcards and other ID components need to be revealed in the ciphertexts.

We summarize the comparison of major different schemes in Table 1.

2 Preliminaries

2.1 Bilinear Maps

We assume that there is an efficient algorithm \mathbf{Gen} for generating bilinear groups. The algorithm \mathbf{Gen} , on input a security parameter 1^κ , outputs a tuple, $\mathbf{G} = [p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, e]$ where $\log_2(p) = \Theta(\kappa)$. A function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map. Here, \mathbb{G} and \mathbb{G}_T are multiplicative groups of prime order p , generated by g and $e(g, g)$ respectively. The bilinear map e has the following properties:

1. Bilinearity: for all $a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

2.2 Complexity Assumptions

We describe complexity assumptions used in our security proofs.

2.2.1 The Decisional Bilinear Diffie-Hellman (DBDH) Assumption

We use the decisional version of the bilinear DH assumption [4, 13]. Let $z_1, z_2, z_3, z \in \mathbb{Z}_p^*$ be chosen at random and $g \in \mathbb{G}$ be a generator. The DBDH assumption is that no probabilistic polynomial-time algorithm can distinguish the tuple $[g, g^{z_1}, g^{z_2}, g^{z_3}, e(g, g)^{z_1 z_2 z_3}]$ from the tuple $[g, g^{z_1}, g^{z_2}, g^{z_3}, e(g, g)^z]$ with non-negligible advantage.

2.2.2 The Decision Linear (D-Linear) Assumption

The D-Linear assumption was first proposed in [3]. Let $z_1, z_2, z_3, z_4, z \in \mathbb{Z}_p^*$ be chosen at random and $g \in \mathbb{G}$ be a generator. The D-Linear assumption is that no probabilistic polynomial-time algorithm can distinguish the tuple $[g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^{z_3 + z_4}]$ from the tuple $[g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^z]$ with non-negligible advantage.

2.3 Access Structure for Ciphertext

In the CP-ABE scheme, an encryptor specifies an access structure for a ciphertext, which is called a ciphertext policy. If a decryptor has a secret key whose associated set of attributes satisfies the access structure, she can decrypt the ciphertext. The access structures used in [2] are the most flexible and expressive. For example, we can use an access structure such as $((A \text{ AND } B) \text{ OR } (C \text{ AND } D))$ in [2]. This means that a recipient must have attributes A and B simultaneously or attributes C and D simultaneously in order to decrypt the ciphertext. Therefore, if a recipient has a secret key associated with a set of attributes $\{A, B, C\}$, she can satisfy the access structure and decrypt the ciphertext. However, if the recipient has a secret key associated with a set of attributes $\{A, C\}$, she can not satisfy the access structure or decrypt the ciphertext. Actually **AND**, **OR**, and threshold gates can be used for expressing the access structures in [2].

However, the security proof of [2] is in the *generic group model*. In order to obtain a reduction-based security proof, Cheung and Newport proposed another CP-ABE scheme [11] which is proved to be secure under *standard* complexity assumptions. The price of obtaining such security proofs is that the expressiveness of ciphertext policies in [11] is somewhat restricted as compared with [2]. However, the expressiveness is not too restrictive and still remains useful.

The access structure and the attribute set associated with the secret key used in [11] are as follows. Let's assume that the total number of attributes in the system is n and the attributes are indexed as $\{\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_i, \dots, \mathbb{A}_n\}$ or we may use just i to refer to \mathbb{A}_i . We use the notation such as $L = [L_1, \dots, L_n] = [1, 0, 1, \dots, 0]$ in order to describe attribute-value pairs for a user, which we call the attribute list. For example, the user has the value 1 for \mathbb{A}_1 , 0 for \mathbb{A}_2 , 1 for \mathbb{A}_3 , \dots , and 0 for \mathbb{A}_n in this case. A trusted authority generates a secret key for the user based on the user's attribute list.

In order to specify the access structure for a ciphertext, we use the notation such as $W = [W_1, \dots, W_n] = [1, 1, *, *, 0]$ where $n = 5$, which we call the ciphertext policy. The wildcard $*$ can be used in the ciphertext policies and it plays the role of "don't care" value. This can be considered as an **AND**-gate on all the attributes. For example, the above ciphertext policy means that the recipient who wants to decrypt must have the value 1 for \mathbb{A}_1 and \mathbb{A}_2 and 0 for \mathbb{A}_5 , and the values for \mathbb{A}_3 and \mathbb{A}_4 do not matter in the **AND**-gate. If the recipient has the secret key associated with, let us say, $[1, 1, 1, 0, 0]$, she can decrypt the ciphertext, but not if the secret key is associated with $[1, 1, 1, 0, 1]$.

Formally, given an attribute list $L = [L_1, L_2, \dots, L_n]$ and a ciphertext policy $W = [W_1, W_2, \dots, W_n]$, L satisfies W if, for all $i = 1, \dots, n$, $L_i = W_i$ or $W_i = *$, and otherwise L does not satisfy W . We use the notation $L \models W$ to mean that L satisfies W .

In our constructions, we generalize the access structures in [11]. In [11], each attribute can take two values 1 and 0, but in our generalized access structures each attribute can take two or more values and each W_i in W can be any subset of possible values for \mathbb{A}_i . More formally, let $S_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,t}, \dots, v_{i,n_i}\}$ be a set of possible values for \mathbb{A}_i where n_i is the number of the possible values for \mathbb{A}_i . Then the attribute list L for a user is $L = [L_1, L_2, \dots, L_i, \dots, L_n]$ where $L_i \in S_i$ and the generalized ciphertext policy W is $W = [W_1, W_2, \dots, W_i, \dots, W_n]$ where $W_i \subseteq S_i$. The generalized ciphertext policy W means, let us say,

$$\begin{aligned} &(\mathbb{A}_1 = v_{1,1} \vee \mathbb{A}_1 = v_{1,3}) \\ &\wedge (\mathbb{A}_2 = v_{2,2}) \wedge \dots \\ &\wedge (\mathbb{A}_i = v_{i,5} \vee \dots \vee \mathbb{A}_i = v_{i,n_i}) \wedge \dots \\ &\wedge (\mathbb{A}_n = v_{n,1} \vee \mathbb{A}_n = v_{n,2} \vee \mathbb{A}_n = v_{n,3}). \end{aligned}$$

When the encryptor specifies a wildcard for \mathbb{A}_i , it corresponds to specifying $W_i = S_i$ for \mathbb{A}_i . The attribute list L satisfies the ciphertext policy W iff $L_i \in W_i$ for $1 \leq i \leq n$. We achieve recipient anonymity by hiding what subset W_i for each \mathbb{A}_i is specified in the access structure of the **AND**-gate of all the attributes.

2.4 Syntax of CP-ABE

Our CP-ABE schemes consist of the following four algorithms.

Setup(1^κ). This algorithm takes the security parameter κ as input and generates a public key PK and a master secret key MK .

KeyGen(MK, L). This algorithm takes MK and an attribute list L as input and generates a secret key SK_L associated with L .

Encrypt(PK, M, W). This algorithm takes PK , a message M , and an ciphertext policy W as input, and generates a ciphertext CT .

Decrypt(CT, SK_L). This algorithm takes CT and SK_L associated with L as input and returns the message M if the attribute list L satisfies the ciphertext policy W specified for CT , that is, $L \models W$. If $L \not\models W$, it returns \perp with overwhelming probability.

2.5 Security Model

We describe the security models for our CP-ABE. Based on [21, 5, 15], we use the following security game. A CP-ABE scheme is selectively secure if no probabilistic polynomial-time adversary has non-negligible advantage in the following game.

Selective Game for CP-ABE

Init: The adversary commits to the challenge ciphertext policies W_0, W_1 .

Setup: The challenger runs the **Setup** algorithm and gives PK to the adversary.

Phase 1: The adversary submits the attribute list L for a **KeyGen** query. If ($L \models W_0 \wedge L \models W_1$) or ($L \not\models W_0 \wedge L \not\models W_1$), the challenger gives the adversary the secret key SK_L . The adversary can repeat this polynomially many times.

Challenge: The adversary submits messages M_0, M_1 to the challenger. If the adversary obtained the SK_L whose associated attribute list L satisfies both W_0 and W_1 in Phase 1, then it is required that $M_0 = M_1$. The challenger flips a random coin b and passes the ciphertext **Encrypt**(PK, M_b, W_b) to the adversary.

Phase 2: Phase 1 is repeated. If $M_0 \neq M_1$, the adversary cannot submit L such that $L \models W_0 \wedge L \models W_1$.

Guess: The adversary outputs a guess b' of b .

The advantage of an adversary in this game is defined as $|\Pr[b' = b] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary. Since the adversary must commit to the challenge ciphertext policies before the setup phase, this model can be considered to be analogous to the selective-ID model [8, 9] used in identity-based encryption schemes. The non-selective-ID model can be found in [2] where the proof is in the generic group

model. In the game, the adversary can submit L such that $L \models W_0$ and $L \models W_1$ if possible and then the adversary can decrypt the ciphertext. This definition captures that the adversary cannot obtain the useful information about the ciphertext policy more than the fact that she can decrypt the ciphertext. The above notion of security is called *match-concealing security* in [21].

3 Proposed Schemes

We construct two CP-ABE schemes that achieve recipient anonymity. In [11], the ciphertext policy needs to be revealed in the ciphertext so that a decryptor can know which secret key components should be used. Furthermore, in order to support wildcards for ciphertext policies, the public key components for the wildcards are prepared in [11] and the decryptor uses the secret key components corresponding to the wildcards if the wildcards are specified in the ciphertext policies. In our constructions, we can hide how the ciphertext policy is specified successfully.

First we show the construction of [11] and later explain the intuition behind our approach we take to make it recipient-anonymous. We assume, for notational simplicity, that the total number of attributes in the system is n and the attributes are indexed as $\{1, 2, \dots, i, \dots, n\}$.

3.1 Construction of [11]

The four algorithms are as follows:

Setup(1^κ). A trusted authority generates a tuple $\mathbf{G} = [p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, \mathbf{e}] \leftarrow \mathbf{Gen}(1^\kappa)$, and random $w \in \mathbb{Z}_p^*$. For each attribute i where $1 \leq i \leq n$, the authority generates random values $a_i, \hat{a}_i, a_i^* \in \mathbb{Z}_p^*$. The authority computes $Y = \mathbf{e}(g, g)^w$ and $A_i = g^{a_i}, \hat{A}_i = g^{\hat{a}_i}, A_i^* = g^{a_i^*}$. The public key PK consists of $\langle Y, p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}, \{A_i, \hat{A}_i, A_i^*\}_{1 \leq i \leq n} \rangle$. The master secret key MK is $\langle w, \{a_i, \hat{a}_i, a_i^*\}_{1 \leq i \leq n} \rangle$.

KeyGen(MK, L). Let $L = [L_1, L_2, \dots, L_n]$ be the attribute list for the user who will obtain the corresponding secret key. The trusted authority picks up random values $s_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq n$, sets $s = \sum_{i=1}^n s_i$, and computes $D_0 = g^{w-s}$. For $1 \leq i \leq n$, the authority also computes $[D_i, D_i^*] = [g^{s_i/a_i}, g^{s_i/a_i^*}]$ if $L_i = 1$, and $[D_i, D_i^*] = [g^{s_i/\hat{a}_i}, g^{s_i/a_i^*}]$ if $L_i = 0$. The secret key SK_L is $\langle D_0, \{D_i, D_i^*\}_{1 \leq i \leq n} \rangle$.

Encrypt(PK, M, W). An encryptor encrypts a message $M \in \mathbb{G}_T$ under a ciphertext policy $W = [W_1, W_2, \dots, W_n]$. The encryptor picks up a random value $r \in \mathbb{Z}_p^*$ and sets $\tilde{C} = MY^r$ and $C_0 = g^r$. Also for $1 \leq i \leq n$, the encryptor computes C_i as follows: if $W_i = 1$, $C_i = A_i^r$; if $W_i = 0$, $C_i = \hat{A}_i^r$; if $W_i = *$, $C_i = A_i^{*r}$. The ciphertext CT is $\langle \tilde{C}, C_0, \{C_i\}_{1 \leq i \leq n} \rangle$. The encryptor needs to reveal W in CT so that recipients can know which secret key components should be used for each C_i .

Note that if W is hidden in CT , the recipients need to try all the possible combinations of the secret key components for decryption and it takes exponential time in n , which seems impractical.

Decrypt(CT, SK_L). The recipient can check W to know whether $L \models W$. If $L \models W$, she can proceed. The recipient decrypts the $CT, \langle \tilde{C}, C_0, \{C_i\}_{1 \leq i \leq n} \rangle$ by using her $SK_L, \langle D_0, \{D_i, D_i^*\}_{1 \leq i \leq n} \rangle$ associated with the attribute list L , as follows:

1. For $1 \leq i \leq n$,

$$D'_i = \begin{cases} D_i & \text{if } W_i \neq * \\ D_i^* & \text{if } W_i = *. \end{cases}$$

- 2.

$$M = \frac{\tilde{C}}{e(C_0, D_0) \prod_{i=1}^n e(C_i, D'_i)}.$$

3.2 Main Idea

We describe how to make the construction of [11] recipient-anonymous. To achieve our goal, the recipients need to be able to decrypt CT without knowing W and we also want to support wildcards in a hidden way. For that, we remove the public key components $\{A_i^*\}_{1 \leq i \leq n}$ for the wildcards and the secret key components $\{D_i^*\}_{1 \leq i \leq n}$ are not included in SK_L . Furthermore, instead of the ciphertext components $\{C_i\}_{1 \leq i \leq n}$, $\{C_i, \hat{C}_i\}_{1 \leq i \leq n}$ are generated with $C_0 = g^r$ as follows: let $\{C_i, \hat{C}_i\} = \{A_i^{r_1}, \hat{A}_i^{r_2}\}$; if $W_i = 1$, we set $r_1 = r$ and r_2 is random; if $W_i = 0$, r_1 is random and $r_2 = r$; if $W_i = *$, $r_1 = r_2 = r$. That is, if $C_i = A_i^r$ or $\hat{C}_i = \hat{A}_i^r$, these ciphertext components are “well-formed” and can be used for successful decryption and otherwise “malformed” (or random). Each decryptor uses C_i for decryption if $L_i = 1$ and uses \hat{C}_i if $L_i = 0$ without knowing what is specified for W_i . By generating the ciphertext like this, we can realize the functionality of wildcards. We can generalize this idea to adapt to the multi-valued attribute setting.

Finally to make it hard to distinguish the well-formed components from the malformed components, we use the linear splitting technique in [6, 21] and make our first construction provably secure as shown in Sect. 4.

3.3 Our First Construction

The four algorithms are as follows:

Setup(1^κ). A trusted authority generates a tuple $\mathbf{G} = [p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, \mathbf{e}] \leftarrow \mathbf{Gen}(1^\kappa)$ and random $w \in \mathbb{Z}_p^*$. For each attribute i where $1 \leq i \leq n$, the authority generates random values $\{a_{i,t}, b_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ and random points $\{A_{i,t} \in \mathbb{G}\}_{1 \leq t \leq n_i}$ [1]. The authority computes $Y = \mathbf{e}(g, g)^w$. The public key

¹ In the asymmetric bilinear groups, $A_{i,t}$ must be generated such that $A_{i,t} = g^{c_{i,t}}$ where $c_{i,t} \in_R \mathbb{Z}_p^*$ and $c_{i,t}$ is known to the authority so that the authority can use $c_{i,t}$ in **KeyGen**.

PK consists of $\langle Y, p, \mathbb{G}, \mathbb{G}_T, g, e, \{\{A_{i,t}^{a_{i,t}}, A_{i,t}^{b_{i,t}}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$. The master secret key MK is $\langle w, \{\{a_{i,t}, b_{i,t}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$.

KeyGen(MK, L). Let $L = [L_1, L_2, \dots, L_n] = [v_{1,t_1}, v_{2,t_2}, \dots, v_{n,t_n}]$ be the attribute list for the user who obtains the corresponding secret key. The trusted authority picks up random values $s_i, \lambda_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq n$, sets $s = \sum_{i=1}^n s_i$, and computes $D_0 = g^{w-s}$. For $1 \leq i \leq n$, the authority also computes $[D_{i,0}, D_{i,1}, D_{i,2}] = [g^{s_i}(A_{i,t_i})^{a_{i,t_i} b_{i,t_i} \lambda_i}, g^{a_{i,t_i} \lambda_i}, g^{b_{i,t_i} \lambda_i}]$ where $L_i = v_{i,t_i}$. The secret key SK_L is $\langle D_0, \{\{D_{i,j}\}_{0 \leq j \leq 2}\}_{1 \leq i \leq n} \rangle$.

Encrypt(PK, M, W). An encryptor encrypts a message $M \in \mathbb{G}_T$ under a ciphertext policy $W = [W_1, W_2, \dots, W_n]$. The encryptor picks up a random value $r \in \mathbb{Z}_p^*$ and sets $\tilde{C} = MY^r$ and $C_0 = g^r$. Also for $1 \leq i \leq n$, the encryptor picks up random values $\{r_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ and computes $\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}$ as follows: if $v_{i,t} \in W_i$, $[C_{i,t,1}, C_{i,t,2}] = [(A_{i,t}^{b_{i,t}})^{r_{i,t}}, (A_{i,t}^{a_{i,t}})^{r-r_{i,t}}]$ (*well-formed*); if $v_{i,t} \notin W_i$, $[C_{i,t,1}, C_{i,t,2}]$ are random (*malformed*). The ciphertext CT is $\langle \tilde{C}, C_0, \{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$.

Decrypt(CT, SK_L). The recipient tries decrypting the CT , $\langle \tilde{C}, C_0, \{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$ without knowing W by using her SK_L , $\langle D_0, \{\{D_{i,j}\}_{0 \leq j \leq 2}\}_{1 \leq i \leq n} \rangle$ associated with the attribute list L , as follows:

1. For $1 \leq i \leq n$,

$$[C'_{i,1}, C'_{i,2}] = [C_{i,t_i,1}, C_{i,t_i,2}] \text{ where } L_i = v_{i,t_i}.$$

- 2.

$$M = \frac{\tilde{C} \prod_{i=1}^n e(C'_{i,1}, D_{i,1}) e(C'_{i,2}, D_{i,2})}{e(C_0, D_0) \prod_{i=1}^n e(C_0, D_{i,0})}.$$

If the attribute list L satisfies the hidden ciphertext policy W of the CT , the recipient can decrypt the CT correctly. For the recipient to know whether the decryption was successful without knowing the ciphertext policy W , we can use the technique used in [5] in practice. As in [5], the encryptor picks up a random $k \in \mathbb{G}_T$ and derives two uniform and independent μ -bit symmetric keys (k_0, k_1) from k . The final ciphertext consists of $\langle k_1, \mathbf{Encrypt}(PK, k, W), E_{k_0}(M) \rangle$ where $\mathbf{Encrypt}(PK, k, W)$ is the ciphertext of k encrypted under PK and W , and $E_{k_0}(M)$ is the ciphertext of M encrypted under k_0 by using a symmetric encryption scheme. The recipient can use k_1 to check whether the decryption was successful after decrypting k where the false positive probability is approximately $1/2^\mu$. If successful, the recipient can decrypt M by using k_0 derived from k . The security proof is given in Sect. 4.

3.4 Second Construction with More Flexibility

We can also apply the technique in Sect. 3.2 to [2] and make it recipient-anonymous. With a property inherited from [2], this scheme is more flexible though the security proof is in the generic group model. The scheme in [2] uses

a symmetric bilinear group while we use an asymmetric bilinear group. That is, we assume $\mathbf{Gen}(1^\kappa)$ outputs $\mathbf{G} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \mathbf{e}]$ where $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map. We also use the External Diffie-Hellman (XDH) assumption used in, for example, [3,20,7] to achieve recipient anonymity, which holds on MNT curves [17]. In the XDH assumption, it holds that the Decisional Diffie-Hellman (DDH) problem is hard in \mathbb{G}_1 and this implies that there does not exist an efficiently-computable isomorphism $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The four algorithms are as follows:

Setup(1^κ). A trusted authority generates a tuple $\mathbf{G} = [p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, \mathbf{e}]$. and random $w, \beta \in \mathbb{Z}_p^*$. For each attribute i where $1 \leq i \leq n$, the authority generates random values $\{a_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ and computes points $\{A_{i,t} = g_1^{a_{i,t}}\}_{1 \leq t \leq n_i}$. The authority computes $Y = \mathbf{e}(g_1, g_2)^w$ and $B = g_1^\beta$. The public key PK consists of $\langle Y, B, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathbf{e}, \{\{A_{i,t}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$. The master secret key MK is $\langle w, \beta, \{\{a_{i,t}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$.

KeyGen(MK, L). Let $L = [L_1, L_2, \dots, L_n] = [v_{1,t_1}, v_{2,t_2}, \dots, v_{n,t_n}]$ be the attribute list for the user who obtains the corresponding secret key. The trusted authority picks up random values $s, \lambda_i \in \mathbb{Z}_p^*$ for $1 \leq i \leq n$ and computes $D_0 = g_2^{\frac{w+s}{\beta}}$. For $1 \leq i \leq n$, the authority also computes $[D_{i,1}, D_{i,2}] = [g_2^{s+a_{i,t_i}\lambda_i}, g_2^{\lambda_i}]$ where $L_i = v_{i,t_i}$. The secret key SK_L is $\langle D_0, \{D_{i,1}, D_{i,2}\}_{1 \leq i \leq n} \rangle$.

Encrypt(PK, M, W). An encryptor encrypts a message $M \in \mathbb{G}_T$ under a ciphertext policy $W = [W_1, W_2, \dots, W_n]$. The encryptor picks up a random value $r \in \mathbb{Z}_p^*$ and sets $\tilde{C} = MY^r$ and $C_0 = B^r$. Also for $1 \leq i \leq n$, the encryptor picks up random values $r_i \in \mathbb{Z}_p^*$ such that $r = \sum_{i=1}^n r_i$, sets $C_{i,1} = g_1^{r_i}$ and computes $\{C_{i,t,2}\}_{1 \leq t \leq n_i}$ as follows: if $v_{i,t} \in W_i$, $C_{i,t,2} = A_{i,t}^{r_i}$ (*well-formed*); if $v_{i,t} \notin W_i$, $C_{i,t,2}$ is random (*malformed*). The ciphertext CT is $\langle \tilde{C}, C_0, \{C_{i,1}, \{C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$.

Decrypt(CT, SK_L). The recipient decrypts the CT , $\langle \tilde{C}, C_0, \{C_{i,1}, \{C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$ by using her SK_L , $\langle D_0, \{D_{i,1}, D_{i,2}\}_{1 \leq i \leq n} \rangle$ associated with the attribute list L as follows:

1. For $1 \leq i \leq n$,

$$C'_{i,2} = C_{i,t_i,2} \text{ where } L_i = v_{i,t_i}.$$

- 2.

$$M = \frac{\tilde{C} \prod_{i=1}^n \mathbf{e}(C_{i,1}, D_{i,1})}{\mathbf{e}(C_0, D_0) \prod_{i=1}^n \mathbf{e}(C'_{i,2}, D_{i,2})}.$$

Under the XDH assumption, it is hard to guess from CT what subset W_i the encryptor specified for each attribute \mathbb{A}_i in the ciphertext policy. The security proof will be similar to that of [2] and is omitted due to space limitation. We discuss the flexibility of this scheme in Sect. 5 in more detail.

4 Overview of Security Proofs

We prove that our first scheme is selectively secure under the DBDH assumption and the D-Linear assumption. We will give the high-level arguments of the proofs here and the detailed proofs of the lemmas are given in Appendix [C](#).

Suppose the adversary commits to the challenge ciphertext policies W_0, W_1 at the beginning of the game. We use the notation $W_b = [W_{b,1}, W_{b,2}, \dots, W_{b,i}, \dots, W_{b,n}]$.

The proof uses a sequence of hybrid games to argue that the adversary cannot win the original security game denoted by G with non-negligible probability. We begin by slightly modifying the game G into a game G_0 . Games G and G_0 are the same except for how the challenge ciphertext is generated. In G_0 , if the adversary did not obtain the SK_L whose associated attribute list L is such that $L \models W_0 \wedge L \not\models W_1$, then the challenge ciphertext component \tilde{C} is a random element of \mathbb{G}_T regardless of the random coin b . The rest of the ciphertext is generated as usual. If the adversary obtained the SK_L whose associated attribute list L is such that $L \models W_0 \wedge L \models W_1$ then the challenge ciphertext in G_0 is generated correctly. That is, $G = G_0$ in this case.

Lemma 1 *Under the DBDH assumption, for any polynomial time adversary \mathcal{A} , the difference of advantage of \mathcal{A} in game G and game G_0 is negligible in the security parameter κ .*

Next, we modify G_0 by changing how to generate the ciphertext components $\{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}$ and define a sequence of games as follows.

For $v_{i,t}$ such that $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \in W_{1,i})$ or $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \notin W_{1,i})$, the components $\{C_{i,t,1}, C_{i,t,2}\}$ are generated as in the real scheme through the sequence of all the games.

If there is $v_{i,t}$ such that $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \notin W_{1,i})$ or $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \in W_{1,i})$, the components $\{C_{i,t,1}, C_{i,t,2}\}$ generated properly in game $G_{\ell-1}$ are replaced with the random values in the new modified game G_ℓ regardless of the random coin b . Every time we replace such components $\{C_{i,t,1}, C_{i,t,2}\}$ with the random values, we define a new modified game. We repeat this replacement one by one until we have no component that satisfies $(v_{i,t} \in W_{0,i} \wedge v_{i,t} \notin W_{1,i})$ or $(v_{i,t} \notin W_{0,i} \wedge v_{i,t} \in W_{1,i})$. In the last game of the sequence, the advantage of the adversary is zero because the adversary is given a ciphertext chosen from the same distribution regardless of the random coin b .

By replacing the well-formed ciphertext components in $G_{\ell-1}$ with the random values in G_ℓ in this way, we can embed a D-Linear challenge in the ciphertext such that the distinguisher of $G_{\ell-1}$ and G_ℓ leads to the distinguisher of the D-Linear challenge.

Lemma 2 *Under the D-Linear assumption, for any polynomial time adversary \mathcal{A} , the difference of advantage of \mathcal{A} in game $G_{\ell-1}$ and game G_ℓ is negligible in the security parameter κ .*

By considering the sequence G, G_0, G_1, \dots of games starting with the original game G , no polynomial adversary can win the game G with non-negligible advantage by the lemmas above.

5 Adding Attributes After *Setup*

In our schemes, it is easy to add new possible values $v_{i,t}$'s of each attribute \mathbb{A}_i in the ciphertext policy even after *Setup* is executed, because we have only to add the public key components for the new values of \mathbb{A}_i and the existing public parameters can remain unchanged. That is, the access structure for the ciphertext policy can be extended accordingly though the ciphertext size is also increased. However, in our first scheme, it cannot be done securely to simply add new attributes \mathbb{A}_i 's in the ciphertext policy with the existing public parameters being unchanged after *Setup* is executed and some users already have their secret keys. The reason is as follows. Suppose there are three attributes $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$ in the system when *Setup* is executed and a user obtains her secret key SK_L where the attribute list $L = [L_1, L_2, L_3] = [1, 1, 0]$. After that, a new attribute \mathbb{A}_4 is added in the system and the corresponding public key components for \mathbb{A}_4 are generated and published. Then an encryptor may specify a ciphertext policy $W = [W_1, \dots, W_4] = [*, *, 0, 1]$, requiring the legitimate recipients to have the value 1 for \mathbb{A}_4 . In this case, the user who has the above SK_L can decrypt the ciphertext encrypted under the ciphertext policy W even if she does not have the secret key component for \mathbb{A}_4 , because L satisfies $[W_1, W_2, W_3]$ partially and it enables the user to combine all the secret key components to reconstruct $s = \sum_{i=1}^n s_i$ in the exponent for decryption. The similar situations can also happen in [11, 15, 5, 21] if we consider the setting where new attributes may be added in the ciphertext policy dynamically after *Setup* is executed. As mentioned in [18], we may be able to prepare redundant filler attributes reserved for future use, but it increases the ciphertext size unnecessarily.

The second scheme can avoid this situation with the property inherited from [2] and we can add new attributes in the ciphertext policy securely after *Setup* is executed where the existing public parameters can remain unchanged. Note that in this scheme, the encryptor splits random r in the ciphertext CT such that $r = \sum_{i=1}^n r_i$ and it forces decryptors to have the secret key components for all the attributes specified in the ciphertext policy even if the attributes in the ciphertext policy were added after the decryptors obtained their secret keys. If a user wants to decrypt the ciphertext with the ciphertext policy including newly added attributes, she must obtain a new secret key including the newly added attributes from the trusted authority again.

Additionally, in the second scheme, an encryptor can specify a variable-length ciphertext policy. For example, the encryptor can specify the ciphertext policy $W = [W_{i_1}, W_{i_2}, \dots, W_{i_m}]$ where $m < n$ and n is the number of all the attributes in the system. Since there are several attributes that do not appear in the ciphertext policy, the partial information on the ciphertext policy is leaked. That is, it means that the wildcards are specified for the attributes not appearing in

the ciphertext policy. However, it may be acceptable to the encryptor in some cases and it can reduce the size of the ciphertext.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments.

References

1. Abdalla, M., Catalano, D., Dent, A., Malone-Lee, J., Neven, G., Smart, N.: Identity-based encryption gone wild. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 300–311. Springer, Heidelberg (2006)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proc. IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
3. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
4. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
6. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
7. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
10. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
11. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: Proc. ACM Conference on Computer and Communications Security (CCS), pp. 456–465 (2007)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proc. ACM Conference on Computer and Communications Security (CCS), pp. 89–98 (2006)
13. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
14. Kapadia, A., Tsang, P.P., Smith, S.W.: Attribute-based publishing with hidden credentials and hidden policies. In: Proc. Network & Distributed System Security Symposium (NDSS), pp. 179–192 (2007)

15. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Eurocrypt 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
Cryptology ePrint Archive 2007/404 (2007)
16. Lubicz, D., Sirvent, T.: Attribute-based broadcast encryption scheme made efficient. In: Africacrypt 2008. LNCS, vol. 5023. Springer, Heidelberg (to appear, 2008)
17. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reductions. IEICE Trans., Fundamentals E84-A(5), 1234–1243 (2001)
18. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proc. ACM Conference on Computer and Communications Security (CCS), pp. 195–203 (2007)
19. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
20. Scott, M.: Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive 2002/164 (2002)
21. Shi, E., Bethencourt, J., Chan, H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Proc. IEEE Symposium on Security and Privacy, pp. 350–364 (2007)

A Realization of CP-ABE with [5]

We show how the scheme in [5] can realize the access structure of the ciphertext policy considered in this work by using HVE. For ease of exposition, suppose there are two attributes $\mathbb{A}_1, \mathbb{A}_2$ in the system and \mathbb{A}_1 can take values $v_{1,1}, v_{1,2}$ and \mathbb{A}_2 can take values $v_{2,1}, v_{2,2}, v_{2,3}$. When an encryptor encrypts a message, the encryptor specifies a vector corresponding to $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2}, v_{2,3})$ as a ciphertext policy. For example, if $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2}, v_{2,3}) = (1, 0, 1, 0, 1)$, this means $(\mathbb{A}_1 = v_{1,1}) \wedge (\mathbb{A}_2 = v_{2,1} \vee \mathbb{A}_2 = v_{2,3})$. A decryptor with $\mathbb{A}_1 = v_{1,1} \wedge \mathbb{A}_2 = v_{2,3}$ obtains her secret key the vector of which corresponds to $(1, *, *, *, 1)$. The decryptor can decrypt the ciphertext if the vectors of both the ciphertext and the secret key match up except the wildcards. In this scheme, the length of the vectors (5 in the example above) is fixed at the system setup. Therefore, the numbers of both attributes and possible values for each attribute specified in the ciphertext policy are fixed at the system setup.

B Realization of CP-ABE with [15]

We show how the scheme in [15] can realize the access structure of the ciphertext policy considered in this work by using the dual of the predicate corresponding polynomial evaluation. Similarly, for ease of exposition, suppose there are two attributes $\mathbb{A}_1, \mathbb{A}_2$ in the system and \mathbb{A}_1 can take values $v_{1,1}, v_{1,2}$ and \mathbb{A}_2 can take values $v_{2,1}, v_{2,2}, v_{2,3}$. In this scheme, decryption succeeds if the vector for the ciphertext (a_1, a_2, \dots, a_n) and the vector for the secret key (x_1, x_2, \dots, x_n) satisfy the condition that $\sum_{i=1}^n a_i x_i = 0$.

When an encryptor encrypts a message with the ciphertext policy $(\mathbb{A}_1 = v_{1,1}) \wedge (\mathbb{A}_2 = v_{2,1} \vee \mathbb{A}_2 = v_{2,3})$, she prepares two polynomials $f_1(x) = c_1 x + c_0$

and $f_2(x) = d_2x^2 + d_1x + d_0$ such that $f_1(v_{1,1}) = 0$, $f_2(v_{2,1}) = 0$ and $f_2(v_{2,3}) = 0$ and specifies the vector $(c_1, c_0, d_2, d_1, d_0)$ as the ciphertext policy. A decryptor with $\mathbb{A}_1 = v_{1,1} \wedge \mathbb{A}_2 = v_{2,3}$ obtains her secret key the vector of which corresponds to $(v_{1,1}, 1, v_{2,3}^2, v_{2,3}, 1)$. For example, when the encryptor specifies a wildcard for attribute \mathbb{A}_2 in the ciphertext policy, she simply uses $f_2(x) = 0$ where $d_2 = d_1 = d_0 = 0$. In this scheme, the length of the vectors (5 in the example above) is fixed at the system setup. Therefore, the maximum size of the subset of attribute values for each attribute specified in the ciphertext policy for successful decryption is fixed at the system setup. Also, the number of attributes specified in the ciphertext policy is fixed at the system setup. However, when the number of possible attribute values is huge and the maximum size of the subset of attribute values specified in the ciphertext policy is small, the scheme in [15] is more advantageous than ours because it can enjoy the smaller ciphertext size and still realize the wildcard functionality.

C Proofs of Lemmas

C.1 Proof of Lemma [□](#)

Proof. We prove our lemma by assuming that a polynomial adversary \mathcal{A} has non-negligible difference ϵ between its advantage in game G and its advantage in game G_0 . We build a simulator \mathcal{B} that can play the DBDH game with advantage ϵ .

Given a DBDH challenge $[g, g^{z_1}, g^{z_2}, g^{z_3}, Z]$ by the challenger where Z is either $e(g, g)^{z_1 z_2 z_3}$ or random with equal probability, the simulator \mathcal{B} creates the following simulation.

Init: The simulator \mathcal{B} runs \mathcal{A} . \mathcal{A} gives \mathcal{B} two challenge ciphertext policies $W_0 = [W_{0,1}, \dots, W_{0,n}]$, $W_1 = [W_{1,1}, \dots, W_{1,n}]$. Then \mathcal{B} flips a random coin $b \in \{0, 1\}$.

Setup: To provide a public key PK to \mathcal{A} , \mathcal{B} sets Y to $e(g, g)^{z_1 z_2}$. This implies $w = z_1 z_2$. For each attribute i where $1 \leq i \leq n$, \mathcal{B} generates $\{A_{i,t}\}_{1 \leq t \leq n_i}$ such that $A_{i,t} = g^{\alpha_{i,t}}$ if $v_{i,t} \in W_{b,i}$ and $A_{i,t} = g^{z_1 \alpha_{i,t}}$ if $v_{i,t} \notin W_{b,i}$ where $\{\alpha_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ are random. Then \mathcal{B} publishes public parameters as in the real scheme by picking up $\{a_{i,t}, b_{i,t}\}_{1 \leq t \leq n_i}$ at random for $1 \leq i \leq n$.

Phase 1: \mathcal{A} submits an attribute list $L = [L_1, \dots, L_n]$ in a secret key query. We consider only the case where $L \not\models W_0 \wedge L \not\models W_1$. The reason for this is by our definition if $L \models W_0 \wedge L \models W_1$, then the challenge messages M_0, M_1 will be equal. In this case, the games G and G_0 are the same, so there is no difference of advantage of \mathcal{A} in G and G_0 . Therefore, \mathcal{B} simply aborts and takes a random guess.

When $L \not\models W_0 \wedge L \not\models W_1$, there must be $k \in \{1, \dots, n\}$ such that $L_k (= v_{k,t_k}) \notin W_{b,k}$.

For $1 \leq i \leq n$, \mathcal{B} picks up $s'_i \in \mathbb{Z}_p^*$ at random. It then sets $s_k = z_1 z_2 + s'_k$ and for every $i \neq k$, sets $s_i = s'_i$. Finally it sets $s = \sum_{i=1}^n s_i = z_1 z_2 + \sum_{i=1}^n s'_i$.

The D_0 component of the secret key can be computed as

$$D_0 = g^{w-s} = g^{z_1 z_2 - s} = g^{-\sum_{i=1}^n s'_i}.$$

For k , \mathcal{B} computes the components $[D_{k,0}, D_{k,1}, D_{k,2}] = [g^{s_k} (A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k}, g^{a_{k,t_k} \lambda_k}, g^{b_{k,t_k} \lambda_k}]$ as follows:

$$\begin{aligned} D_{k,0} &= g^{s_k} (A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{z_1 z_2 + s'_k} (A_{k,t_k})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{z_1 z_2 + s'_k} (g^{z_1 \alpha_{k,t_k}})^{a_{k,t_k} b_{k,t_k} \lambda_k} \\ &= g^{s'_k} (g^{z_1 \alpha_{k,t_k}})^{a_{k,t_k} b_{k,t_k} \lambda'_k} \end{aligned}$$

where λ_k is chosen at random such that

$$\lambda_k = -\frac{z_2}{\alpha_{k,t_k} a_{k,t_k} b_{k,t_k}} + \lambda'_k$$

and random λ'_k is known to \mathcal{B} .

\mathcal{B} can compute the components $[D_{k,1}, D_{k,2}]$ easily.

For $i \neq k$, \mathcal{B} can also compute $[D_{i,0}, D_{i,1}, D_{i,2}]$ easily.

Challenge: \mathcal{A} submits two challenge messages M_0 and M_1 . \mathcal{B} sets $\tilde{C} = M_b Z$ and $C_0 = g^{z_3}$ which implies $r = z_3$ and generates, for W_b , the ciphertext $\langle \tilde{C}, C_0, \{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n} \rangle$. When $v_{i,t} \in W_{b,i}$, \mathcal{B} can generate $\{C_{i,t,1}, C_{i,t,2}\}$ correctly because $A_{i,t}$ does not contain unknown z_1 and when $v_{i,t} \notin W_{b,i}$, $\{C_{i,t,1}, C_{i,t,2}\}$ can be simply chosen at random.

Phase 2: Phase 1 is repeated.

Guess: \mathcal{A} outputs a guess b' of b . If $b' = b$, \mathcal{B} outputs 1 and otherwise outputs 0. By our assumption, the probability that \mathcal{A} guesses b correctly in game G has a non-negligible ϵ difference from that of it guessing b correctly in G_0 . When $Z = e(g, g)^{z_1 z_2 z_3}$, \mathcal{A} is in game G and when Z is random, \mathcal{A} is in game G_0 . Therefore the simulator \mathcal{B} has advantage ϵ in the DBDH game. \square

C.2 Proof of Lemma 2

Proof. We prove our lemma by assuming that a polynomial adversary \mathcal{A} has non-negligible difference ϵ between its advantage in game $G_{\ell-1}$ and its advantage in game G_ℓ . We build a simulator \mathcal{B} that can play the D-Linear game with advantage ϵ .

Given a D-Linear challenge $[g, g^{z_1}, g^{z_2}, Z, g^{z_2 z_4}, g^{z_3 + z_4}]$ by the challenger where Z is either $g^{z_1 z_3}$ or random with equal probability, the simulator \mathcal{B} creates the simulation. Note that this D-Linear assumption is equivalent to that of Sect.

2.2.2

As mentioned in Sect. 4 in $G_{\ell-1}$, the ciphertext components $\{C_{i_\ell, t_\ell, 1}, C_{i_\ell, t_\ell, 2}\}$ are generated as in the real scheme, whereas, in G_ℓ , the components are random regardless of the random coin b and we assume that $(v_{i_\ell, t_\ell} \in W_{1, i_\ell} \wedge v_{i_\ell, t_\ell} \notin W_{0, i_\ell})$ without loss of generality.

Init: The simulator \mathcal{B} runs \mathcal{A} . \mathcal{A} gives \mathcal{B} two challenge ciphertext policies $W_0 = [W_{0,1}, \dots, W_{0,n}], W_1 = [W_{1,1}, \dots, W_{1,n}]$. Then \mathcal{B} flips a random coin $b \in \{0, 1\}$. If $b = 0$, \mathcal{B} aborts and takes a random guess. The reason for this is by our definition if $b = 0$ where $(v_{i_\ell, t_\ell} \in W_{1,i} \wedge v_{i_\ell, t_\ell} \notin W_{0,i})$, we have $G_{\ell-1} = G_\ell$ because the distribution of the challenge ciphertext in game $G_{\ell-1}$ is the same as that of game G_ℓ , so there is no difference of advantage of \mathcal{A} in $G_{\ell-1}$ and G_ℓ . We proceed assuming $b = 1$.

Setup: To provide a public key PK to \mathcal{A} , \mathcal{B} sets Y to $e(g, g)^w$ where w is known to \mathcal{B} . For each attribute i where $1 \leq i \leq n$, \mathcal{B} generates $\{A_{i,t}\}_{1 \leq t \leq n_i}$ such that $A_{i,t} = g^{\alpha_{i,t}}$ if $v_{i,t} \in W_{b,i}$ and $A_{i,t} = g^{z_1 \alpha_{i,t}}$ if $v_{i,t} \notin W_{b,i}$ where $\{\alpha_{i,t} \in \mathbb{Z}_p^*\}_{1 \leq t \leq n_i}$ are random. Then \mathcal{B} publishes public parameters as in the real scheme by picking up $\{a_{i,t}, b_{i,t}\}_{1 \leq t \leq n_i}$ at random for $1 \leq i \leq n$ with the exception that, for a_{i_ℓ, t_ℓ} and b_{i_ℓ, t_ℓ} , \mathcal{B} sets $a_{i_\ell, t_\ell} = z_1$ and $b_{i_\ell, t_\ell} = z_2$ and can compute $A_{i_\ell, t_\ell}^{a_{i_\ell, t_\ell}} = g^{\alpha_{i_\ell, t_\ell} a_{i_\ell, t_\ell}}$ and $A_{i_\ell, t_\ell}^{b_{i_\ell, t_\ell}} = g^{\alpha_{i_\ell, t_\ell} b_{i_\ell, t_\ell}}$ without knowing z_1, z_2 .

Phase 1: \mathcal{A} submits an attribute list $L = [L_1, \dots, L_n]$ in a secret key query. If $L_{i_\ell} \neq v_{i_\ell, t_\ell}$, \mathcal{B} can generate the corresponding secret key easily. Let's assume $L_{i_\ell} = v_{i_\ell, t_\ell}$. \mathcal{B} needs to compute the secret key components $[D_{i_\ell, 0}, D_{i_\ell, 1}, D_{i_\ell, 2}] = [g^{s_{i_\ell}} (A_{i_\ell, t_\ell})^{a_{i_\ell, t_\ell} b_{i_\ell, t_\ell} \lambda_{i_\ell}}, g^{a_{i_\ell, t_\ell} \lambda_{i_\ell}}, g^{b_{i_\ell, t_\ell} \lambda_{i_\ell}}]$ where $a_{i_\ell, t_\ell} = z_1, b_{i_\ell, t_\ell} = z_2$. \mathcal{B} can compute $D_{i_\ell, 0}$ as

$$\begin{aligned} D_{i_\ell, 0} &= g^{s_{i_\ell}} (A_{i_\ell, t_\ell})^{a_{i_\ell, t_\ell} b_{i_\ell, t_\ell} \lambda_{i_\ell}} \\ &= g^{s_{i_\ell}} (A_{i_\ell, t_\ell})^{z_1 z_2 \lambda_{i_\ell}} \\ &= g^{s_{i_\ell}} (g^{\alpha_{i_\ell, t_\ell}})^{z_1 z_2 \lambda_{i_\ell}} \\ &= g^{s'_{i_\ell}} \end{aligned}$$

where s_{i_ℓ} is chosen at random such that

$$s_{i_\ell} = s'_{i_\ell} - \alpha_{i_\ell, t_\ell} z_1 z_2 \lambda_{i_\ell}$$

and random s'_{i_ℓ} is known to \mathcal{B} . \mathcal{B} can compute the components $[D_{i_\ell, 1}, D_{i_\ell, 2}]$ easily without knowing z_1, z_2 .

Here we can assume $L \not\in W_0 \wedge L \not\in W_1$ because $L_{i_\ell} = v_{i_\ell, t_\ell} \wedge v_{i_\ell, t_\ell} \notin W_{1-b, i_\ell}$. That is, we have $L \not\in W_{1-b}$ and therefore $L \not\in W_b$, so there must be $k \in \{1, \dots, n\}$ such that $L_k (= v_{k, t_k}) \notin W_{b, k}$. Then \mathcal{B} generates $[D_{k, 0}, D_{k, 1}, D_{k, 2}]$ as follows: \mathcal{B} sets $s_k = s'_k + \alpha_{i_\ell, t_\ell} z_1 z_2 \lambda_{i_\ell}$ where s'_k is random and known to \mathcal{B} and computes

$$\begin{aligned} D_{k, 0} &= g^{s_k} (A_{k, t_k})^{a_{k, t_k} b_{k, t_k} \lambda_k} \\ &= g^{s'_k + \alpha_{i_\ell, t_\ell} z_1 z_2 \lambda_{i_\ell}} (g^{z_1 \alpha_{k, t_k}})^{a_{k, t_k} b_{k, t_k} \lambda_k} \\ &= g^{s'_k} (g^{z_1 \alpha_{k, t_k}})^{a_{k, t_k} b_{k, t_k} \lambda'_k} \end{aligned}$$

where λ_k is chosen at random such that

$$\lambda_k = \lambda'_k - \frac{\alpha_{i_\ell, t_\ell} z_2 \lambda_{i_\ell}}{\alpha_{k, t_k} a_{k, t_k} b_{k, t_k}}$$

and random λ'_k is known to \mathcal{B} . \mathcal{B} can compute the components $[D_{k,1}, D_{k,2}]$ easily without knowing z_2 .

Also, for $i \neq i_\ell, k$, \mathcal{B} can compute $[D_{i,0}, D_{i,1}, D_{i,2}]$ easily.

Finally by computing

$$\begin{aligned} s &= \sum_{i=1}^n s_i \\ &= s_{i_\ell} + s_k + \sum_{i \neq i_\ell, k} s_i \\ &= s'_{i_\ell} - \alpha_{i_\ell, t_\ell} z_1 z_2 \lambda_{i_\ell} + s'_k + \alpha_{i_\ell, t_\ell} z_1 z_2 \lambda_{i_\ell} + \sum_{i \neq i_\ell, k} s_i \\ &= s'_{i_\ell} + s'_k + \sum_{i \neq i_\ell, k} s_i, \end{aligned}$$

the component $D_0 = g^{w-s}$ of the secret key can be computed.

Challenge: \mathcal{A} submits two challenge messages M_0 and M_1 . \mathcal{B} sets $C_0 = g^{z_3+z_4}$ which implies $r = z_3 + z_4$. If $L \not\equiv W_0 \wedge L \not\equiv W_1$ for every queried L , \mathcal{B} sets \tilde{C} to be random and otherwise sets $\tilde{C} = M_b e(g, g^{z_3+z_4})^w$. \mathcal{B} generates, for W_b , the ciphertext components $\{\{C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n}$ as in $G_{\ell-1}$ with the exception that the components $\{C_{i_\ell, t_\ell, 1}, C_{i_\ell, t_\ell, 2}\}$ are computed as

$$\begin{aligned} C_{i_\ell, t_\ell, 1} &= (A_{i_\ell, t_\ell}^{b_{i_\ell, t_\ell}})^{r_{i_\ell, t_\ell}} = (A_{i_\ell, t_\ell}^{z_2})^{z_4} = (g^{\alpha_{i_\ell, t_\ell} z_2})^{z_4}, \\ C_{i_\ell, t_\ell, 2} &= (A_{i_\ell, t_\ell}^{a_{i_\ell, t_\ell}})^{r - r_{i_\ell, t_\ell}} = (g^{\alpha_{i_\ell, t_\ell} z_1})^{z_3} = Z^{\alpha_{i_\ell, t_\ell}} \end{aligned}$$

without knowing $z_2 z_4, z_1 z_3$. This implies that $r_{i_\ell, t_\ell} = z_4$ and $Z = g^{z_1 z_3}$ and if $Z = g^{z_1 z_3}$, the components are well-formed and \mathcal{A} is in game $G_{\ell-1}$.

Phase 2: Phase 1 is repeated.

Guess: \mathcal{A} outputs a guess b' of b . If $b' = b$, \mathcal{B} outputs 1 and otherwise outputs 0. By our assumption, the probability that \mathcal{A} guesses b correctly in game $G_{\ell-1}$ has a non-negligible ϵ difference from that of it guessing b correctly in G_ℓ . When $Z = g^{z_1 z_3}$, \mathcal{A} is in game $G_{\ell-1}$ and when Z is random, \mathcal{A} is in game G_ℓ . Therefore the simulator \mathcal{B} has advantage ϵ in the D-Linear game. \square

Attacking Reduced Round SHA-256

Somitra Kumar Sanadhya* and Palash Sarkar

Applied Statistics Unit,
Indian Statistical Institute,
203, B.T. Road, Kolkata,
India 700108.

somitra_r@isical.ac.in, palash@isical.ac.in

Abstract. The SHA-256 hash function has started getting attention recently by the cryptanalysis community due to the various weaknesses found in its predecessors such as MD4, MD5, SHA-0 and SHA-1. We make two contributions in this work. First we describe message modification techniques and use them to obtain an algorithm to generate message pairs which collide for the actual SHA-256 reduced to 18 steps. Our second contribution is to present differential paths for 19, 20, 21, 22 and 23 steps of SHA-256. We construct parity check equations in a novel way to find these characteristics. Further, the 19-step differential path presented here is constructed by using only 15 local collisions, as against the previously known 19-step near collision differential path which consists of interleaving of 23 local collisions. Our 19-step differential path can also be seen as a single local collision at the message word level. We use a linearized local collision in this work. These results do not cause any threat to the security of the SHA-256 hash function.

1 Introduction

Cryptanalysis of hash functions has been an area of intense interest to the research community since past decade and a half. Many hash functions were broken in this time, most notable among them are MD4, MD5, SHA-0 and theoretical break of SHA-1. This has directed the attention of the cryptology community to the SHA-2 family of hash functions.

Known Results for the SHA-2 Family: Gilbert and Handschuh (GH) [5] were the first to study local collisions in the SHA-2 family. They reported a 9-round local collision and estimated the probability of the differential path to be 2^{-66} . This probability estimate was later improved by [11] and [6]. Sanadhya and Sarkar [16] recently presented 16 new 9-round local collisions for SHA-2 family of hash functions. The message expansion of SHA-256 was studied by Mendel et al. [11], who mentioned an 18-step collision for SHA-256 which was recently corrected in [12]. The work [11] also provided a differential path for 19-step near collision for SHA-256. An earlier work [10] studied a very simplified variant of

* This author is supported by the Ministry of Information Technology, Govt. of India.

SHA-256. The encryption mode of SHA-256 is analyzed in [23] and is not relevant to collision search attacks. Recently, at FSE '08, Nikolić and Biryukov [13] reported 21-step collisions for SHA-256 using a nonlinear differential path.

Our Contributions: We make two independent contributions in this work :

1. We construct a 18-step collision characteristic using one of the local collisions from [16]. We describe message modification techniques to find messages following this differential characteristic. Using these techniques, we provide an algorithm to generate pairs of messages which collide for 18 step SHA-256 with the standard IV. We show two such pairs of messages.
2. We show multiple differential paths for attacking up to 23-step SHA-256. In obtaining these differential paths, we use coding theoretic methods in a novel way. Using linearized local collisions, there were no colliding differential paths known for SHA-256 beyond 18 rounds. Previously known best differential path was for 19-step SHA-256 which used 23 local collisions and gave rise to a near collision. In contrast, our 19-step characteristic uses only 15 local collisions and is an exact collision path. All the 15 local collisions start in the same word and therefore this differential path can also be seen as consisting of a single local collision with the starting word difference having a weight of 15 bits. In addition there are no impossible conditions caused by the f_{IF} and f_{MAJ} functions for the differential paths reported here. Therefore the search for actual colliding message pairs following these paths is likely to be easier.

We also show that neutral bit technique may not be of much help in finding actual colliding pair of messages while message modification methods seem to hold much more promise.

Note that these results do not cause any threat to the security of the SHA-256 hash function since it has 64 steps per block.

2 Notation

In this paper we use the following notation:

- $m_i \in \{0, 1\}^{32}$, $W_i \in \{0, 1\}^{32}$, $W'_i \in \{0, 1\}^{32}$ for any i .
- The colliding message pair is: $\{m_0, m_1, m_2, \dots, m_{15}\}$ and $\{m'_0, m'_1, m'_2, \dots, m'_{15}\}$.
- The expanded message pair is: $\{W_0, W_1, W_2, \dots, W_{63}\}$ and $\{W'_0, W'_1, W'_2, \dots, W'_{63}\}$.
- \oplus : bitwise XOR.
- $+$: addition modulo 2^{32} .
- $\Delta W_i = W_i \oplus W'_i$
- $\text{ROTR}^n(x)$: Right rotation of a 32 bit quantity x by n bits.
- $\text{SHR}^n(x)$: Right shift of a 32 bit quantity x by n bits.

3 The SHA-256 Hash Function

The newest members of SHA family of hash functions were standardized by US NIST in 2002 [18]. There are 2 differently designed functions in this standard: the SHA-256 and SHA-512. In addition, the standard also specifies a truncated version of SHA-512, namely the SHA-384. The number in the name of the hash function refers to the length of message digest produced by that function. In this work we are interested in reduced round collision attacks against SHA-256. Next we briefly describe SHA-256. For details refer to [18].

The round function of SHA-256 hash function uses 8 registers. The initial value in the registers is specified by an 8x32 bit IV. In Step i , the 8 registers are updated from $(a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}, e_{i-1}, f_{i-1}, g_{i-1}, h_{i-1})$ to $(a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i)$ according to the following equations:

$$\left. \begin{aligned} a_i &= \Sigma_0(a_{i-1}) + f_{MAJ}(a_{i-1}, b_{i-1}, c_{i-1}) + \Sigma_1(e_{i-1}) \\ &\quad + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) + h_{i-1} + K_i + W_i \\ b_i &= a_{i-1} \\ c_i &= b_{i-1} \\ d_i &= c_{i-1} \\ e_i &= d_{i-1} + \Sigma_1(e_{i-1}) + f_{IF}(e_{i-1}, f_{i-1}, g_{i-1}) \\ &\quad + h_{i-1} + K_i + W_i \\ f_i &= e_{i-1} \\ g_i &= f_{i-1} \\ h_i &= g_{i-1} \end{aligned} \right\} \quad (1)$$

The f_{IF} and the f_{MAJ} are three variable boolean functions “Choice” and “Majority” respectively. The functions Σ_0 and Σ_1 are defined as:

$$\begin{aligned} \Sigma_0(x) &= ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \\ \Sigma_1(x) &= ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \end{aligned}$$

Round i uses a 32 bit word W_i which is derived from the message and a constant word K_i . There are 64 rounds in all. The hash function operates on a 512 bit message specified as 16 words of 32 bits. Given the message words m_0, m_1, \dots, m_{15} , the W_i 's are computed using the equation:

$$W_i = \begin{cases} m_i & \text{for } 0 \leq i \leq 15 \\ \sigma_1(m_{i-2}) + m_{i-7} + \sigma_0(m_{i-15}) + m_{i-16} & \text{for } 16 \leq i \leq 63 \end{cases} \quad (2)$$

The functions σ_0 and σ_1 are defined as:

$$\begin{aligned} \sigma_0(x) &= ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \\ \sigma_1(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \end{aligned}$$

The IV = $(a_{-1}, b_{-1}, c_{-1}, d_{-1}, e_{-1}, f_{-1}, g_{-1}, h_{-1})$ is defined by 8 32-bit constants. All additions in Equations 1 and 2 are modulo 2^{32} .

The output hash value of a one block (512 bit) message is obtained by chaining the IV with the register values at the end of the final round as per the Merkle-Damgård construction. A similar strategy is used for multi-block messages, where the IV for next block is taken as the hash output of the previous block.

4 Collision Attacks Against Hash Functions

The aim of a hash function attack is to produce two different messages both of which map to the same hash output. This is done by employing differential attack against the hash function in question. First a suitable difference of messages is found such that a pair of messages having that difference is likely to collide to the same hash value with high probability. For example, if a given message differential $\{\Delta W_0, \Delta W_1, \dots, \Delta W_{15}\}$ is likely to generate colliding pairs with probability $\frac{1}{2^8}$ then one needs to try roughly 2^8 different pairs $\{W_0, W_1, \dots, W_{15}\}$ and $\{W'_0, W'_1, \dots, W'_{15}\}$ having the given difference to get a colliding pair of messages.

However, the probability of the specified differential to generate a collision is likely to be very low for most of the practical hash functions. Hence some sophisticated methods are used to search for the right (colliding) pair, rather than generating them at random. Message modification techniques [22,20] and neutral bit technique [1] are the two widely used methods to find colliding message pairs.

For a fuller discussion of linearized local collisions and differential paths, refer to [17]. We next discuss the SHA-256 linearized local collisions.

4.1 Linearized Local Collisions in SHA-256

Let the first step in SHA-2 be denoted by Step 0. If a 9-step local collision is started at step i , it defines the 9 word differences $W_j \oplus W'_j$ for $i \leq j \leq i + 8$. We use two types of local collisions in the present work. The first is due to Gilbert and Handschuh [5] and the second is one of the 16 local collisions presented in [16]. From among the 16, we choose the 5th local collision because of the following two reasons :

1. It is one of the 4 which are suitable for getting 18-step collision, as explained later (the others being 7th, 14th and 16th).
2. It has the highest probability among these 4.

We call the two local collisions the GH local collision and the SS₅ local collision respectively. The other three local collisions from [16] are denoted by SS₇, SS₁₄ and SS₁₆.

The following approximations are used in these local collisions :

1. Operator $+$ is approximated by \oplus .
2. In GH, f_{IF} and f_{MAJ} are approximated by zero function. This causes certain impossible conditions while searching for the message pair following this differential path, as has been observed in [11].
3. In SS₅, f_{IF} and f_{MAJ} are approximated by their middle arguments. These linear approximations avoid two types of impossible conditions encountered when using GH local collision.

See [16] for details on other local collisions.

All the local collisions mentioned above are:

- GH : $\{x, \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(\Sigma_1(x)), 0, x, \Sigma_0(x) \oplus \Sigma_1(x), 0, 0, x\}$
- SS₅ : $\{x, \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(\Sigma_1(x)), \Sigma_0(x) \oplus \Sigma_1(x), 0, \Sigma_0(x) \oplus \Sigma_1(x), 0, 0, x\}$
- SS₇ : $\{x, x \oplus \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(x) \oplus \Sigma_0(\Sigma_1(x)), x \oplus \Sigma_0(x) \oplus \Sigma_1(x), 0, x \oplus \Sigma_0(x) \oplus \Sigma_1(x), 0, 0, x\}$
- SS₁₄ : $\{x, x \oplus \Sigma_0(x) \oplus \Sigma_1(x), x \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x)), \Sigma_1(x), \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(x) \oplus \Sigma_1(x), 0, 0, x\}$
- SS₁₆ : $\{x, \Sigma_0(x) \oplus \Sigma_1(x), \Sigma_0(x) \oplus \Sigma_1(x) \oplus \Sigma_0(\Sigma_1(x)), x \oplus \Sigma_1(x), x \oplus \Sigma_0(x) \oplus \Sigma_1(x), x \oplus \Sigma_0(x) \oplus \Sigma_1(x), 0, 0, x\}$

Note that in all the above local collisions, Σ_0 and Σ_1 are used as operators on 32 bit quantities, and x is any 32 bit message word difference. Once a starting message difference x is chosen, next 8 words must have the difference in accordance with the local collision.

5 Attacking 18 Rounds of SHA-256

It is possible to get up to 18 step reduced round collisions for SHA-256 using a single local collision. Such an idea has already been used in [11] and mentioned in [16]. We describe this for clarity of exposition.

First of all, note that any local collision under consideration spans 9 steps and the message expansion of SHA-256 does not play any role in the first 16 steps. Therefore if a local collision spans from Step i to Step $(i + 8)$, and if we take $\Delta W_0 = \Delta W_1 = \dots = \Delta W_{i-1} = \Delta W_{i+9} = \Delta W_{i+10} = \dots = \Delta W_{15} = 0$, we get a differential path for 16-step collision for SHA-256.

The issue of message expansion is not considered in obtaining the 16 step colliding differential path described above. Next we tackle two steps of the message expansion.

Message expansion rule for W_{16} and W_{17} are given by :

$$W_{16} = \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0 \quad (3)$$

$$W_{17} = \sigma_1(W_{15}) + W_{10} + \sigma_0(W_2) + W_1 \quad (4)$$

Let a local collision \mathcal{L} start at Step 3 and hence end at Step 11. This local collision defines the 9 word differences $\Delta W_3, \Delta W_4, \dots, \Delta W_{11}$. The first step of the local collision corresponds to ΔW_3 and the 9th step corresponds to ΔW_{11} . Taking the differentials of all the message words outside the span of the local collision to be zero, the differential path for \mathcal{L} will have $\Delta W_0 = \Delta W_1 = \Delta W_2 = \Delta W_{12} = \Delta W_{13} = \Delta W_{14} = \Delta W_{15} = 0$.

Note that $\Delta W_i = 0$ means that $W_i = W'_i$. Since $\Delta W_0 = \Delta W_1 = \Delta W_{14} = 0$ for \mathcal{L} , from Equation 3, W_{16} and W'_{16} may be different only due to the differences in W_9 and W'_9 .

ΔW_9 corresponds to the 7th step word difference for \mathcal{L} . If \mathcal{L} is chosen such that its 7th step word difference is zero, then $W_9 = W'_9$. Therefore even after the message expansion recursion is used, we will have $W_{16} = W'_{16}$. This results in a 17-step differential path for SHA-256.

Table 1. 18 step linear characteristic for SHA-256. Only 1 SS₅ local collision is used to build this path.

Step i	ΔW_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i
0-2	0	0	0	0	0	0	0	0	0
3	80000000	80000000	0	0	0	80000000	0	0	0
4	22140240	0	80000000	0	0	20040200	80000000	0	0
5	42851098	0	0	80000000	0	80000000	20040200	80000000	0
6	22140240	0	0	0	80000000	0	80000000	20040200	80000000
7	0	0	0	0	0	80000000	0	80000000	20040200
8	22140240	0	0	0	0	0	80000000	0	80000000
9	0	0	0	0	0	0	80000000	0	0
10	0	0	0	0	0	0	0	0	80000000
11	80000000	0	0	0	0	0	0	0	0
12-17	0	0	0	0	0	0	0	0	0

Similarly, if the 8th message word difference for \mathcal{L} is zero, then by Equation 4, $W_{17} = W'_{17}$. This results in a 18-step differential path for SHA-256.

Both the 17 and the 18 Step paths discussed above use just one local collision. To increase the probability of this differential path for the case of real SHA-256, we can take starting messages differing in only 1 bit.

All the local collisions listed in the previous section have the 7th and the 8th message word differences zero. Therefore any one of them can be used to obtain the 18 step colliding differential path for SHA-256. We list one of these differential paths in Table 1. This 18-step colliding path is also a 17-step colliding path for SHA-256.

Further, it can be seen that it is not possible to obtain a differential path for 19 or more steps with a single local collision where the weight of the perturbation in first word is just 1-bit. This impossibility arises due to the message expansion of SHA-256, and because there are no local collisions in which 3 consecutive word differences are zero. We discuss the case of more than 18 steps in later sections.

6 Message Modification Techniques for SHA-256

We have used XOR differences for registers and message words in the differential path for reduced round SHA-256. The differential path in Table 1 is obtained by using linearized SHA-256. However our aim is to obtain a pair of messages which follows this differential path for real SHA-256. The probability for this to happen for random messages is 2^{-49} for 18-step SHA-256. If the message-pair satisfies certain conditions then the probability of the differential path can be increased significantly. We list conditions on the registers and the message words which help in finding messages following the 18 step differential path shown in Table 1 when actual SHA-256 is used. These conditions try to ensure that the functions f_{IF} and f_{MAJ} both behave like their middle arguments, and that $+$ behaves like \oplus . These conditions are shown in Table 2. Sufficient conditions for 9 step SHA-256 collision have also been given in [7], Table 3. We next highlight the advantages of our conditions with those in [7].

1. The conditions in [7] are for only 9-step collision in SHA-256. Our conditions are for 18-step collision in SHA-256.

Table 2. Conditions for the 18 step differential path in Table 1. x^i denotes i^{th} bit of a 32 bit quantity x . \bar{x} denotes the bitwise negation of x which can be a 32 bit or a 1 bit quantity. Operator $+$ is addition modulo 2^{32} and operator $*$ is multiplication of 2 single bits. Both these operators are used in steps 6 and 8.

Step k	Due to f_{MAJ}	Pr.	Due to f_{IF}	Pr.	Due to a_k	Pr.	Due to e_k	Pr.	Step Pr.
0-3	-	-	-	-	-	-	-	-	1
4	$b_3^{31} = c_3^{31}$	$\frac{1}{2}$	$f_3^{31} = g_3^{31}$	$\frac{1}{2}$	$W_4^i = (\Sigma_0(a_3))^i; i = 9, 18, 29$ $W_4^i = (\Sigma_1(e_3))^i; i = 6, 20, 25$	$\frac{1}{2^6}$	bit differences $\Delta W_4^i; i = 9, 18, 29$ propagate into e_4	$\frac{1}{2^3}$	$\frac{1}{2^{11}}$
5	$a_4^{31} = c_4^{31}$	$\frac{1}{2}$	$e_4^{31} = 1,$ $e_3^i = f_3^i;$ $i = 9, 18, 29$	$\frac{1}{2^4}$	$W_5^i = (\Sigma_1(e_4))^i;$ $i = 3, 4, 7, 12, 16,$ $18, 23, 25, 30$	$\frac{1}{2^9}$	-	-	$\frac{1}{2^{14}}$
6	$a_5^{31} = b_5^{31}$	$\frac{1}{2}$	$e_5^{31} = 1;$ $i = 9, 18, 29$ $e_4^{31} = e_2^{31} * e_3^{31}$	$\frac{1}{2^4}$	$W_5^i = (\Sigma_1(e_5) + f_5)^i;$ $i = 6, 9, 18, 20, 25, 29$	$\frac{1}{2^6}$	-	-	$\frac{1}{2^{11}}$
7	-	-	$e_6^i = 1;$ $i = 9, 18, 29, 31$	$\frac{1}{2^4}$	-	-	-	-	$\frac{1}{2^4}$
8	-	-	$e_6^{31} = e_7^{31} * e_5^{31}$	$\frac{1}{2}$	$W_8^i = (\Sigma_1(e_7) + h_7)^i;$ $i = 6, 9, 18, 20, 25, 29$	$\frac{1}{2^6}$	-	-	$\frac{1}{2^7}$
9	-	-	$e_8^{31} = 1$	$\frac{1}{2}$	-	-	-	-	$\frac{1}{2}$
10	-	-	$e_9^{31} = 1$	$\frac{1}{2}$	-	-	-	-	$\frac{1}{2}$
11-17	-	-	-	-	-	-	-	-	1
Prob.									$\frac{1}{2^{49}}$

- The GH local collision is used in [7] whereas we use SS_5 local collision. Further, no explanation is provided in [7] on how these conditions are derived whereas we provide complete details about our conditions. It is now possible to use the method described in this work to derive conditions for 18-step SHA-256 collision using any other local collision.
- In [7] the conditions are claimed to be “sufficient” but it is not clear if satisfying them will immediately lead to a collision. The conditions that we identify are not claimed to be sufficient. We only note that satisfying them will increase the probability of finding colliding message pairs.

6.1 Explanation of Conditions in Table 2

$\Delta W_k = 0$ for steps $k=0, 1$ and 2 and hence there are no restrictions due to these steps. In Step 3, although $\Delta W_3 \neq 0$, the difference is only in the most significant bit. The $+$ and \oplus behave the same with probability 1 for a difference in MSB, so even Step 3 does not impose any restrictions. Hence conditions are needed to tackle the proper differential behavior for the message pair only from Step 4 onwards.

Conditions Due to f_{MAJ} and f_{IF} : In Step 4, f_{MAJ} has inputs a_3, b_3 and c_3 with $\Delta a_3 = 0x80000000$. In SS_5 local collision f_{MAJ} is approximated by it’s middle argument, which will happen if $b_3^{31} = c_3^{31}$. Similarly the f_{IF} function having arguments e_3, f_3 and g_3 will behave like it’s middle argument if $f_3^{31} = g_3^{31}$.

Conditions Due to Register a_4 : Once the two boolean functions are approximated by their middle arguments, register a_4 is evaluated for both the messages as follows :

$$\begin{aligned}
 a_4 &= \Sigma_0(a_3) + b_3 + \Sigma_1(e_3) + f_3 + h_3 + K_4 + W_4 \quad \text{and} \\
 a'_4 &= \Sigma_0(a'_3) + b'_3 + \Sigma_1(e'_3) + f'_3 + h'_3 + K_4 + W'_4
 \end{aligned}$$

Registers a_3 and a'_3 (resp. e_3 and e'_3) differ in their MSB, and the operator Σ_0 (resp. Σ_1) expands this difference to 3 bit positions 6, 20 and 25 (resp. 9, 18 and 29). The word difference ΔW_4 at this step has been chosen to differ in these 6 bit positions (namely 6, 20, 25, 9, 18 and 29) with the aim of cancelling these differences.

The cancellation will happen as desired if :

1. The difference of words W_4 and W'_4 is opposite to the difference in words $\Sigma_0(a_3)$ and $\Sigma_0(a'_3)$ on bit positions 9, 18 and 29. For example, if $(\Sigma_0(a_3))^i = 1$ and $(\Sigma_0(a'_3))^i = 0$, then we would like $(W_4)^i = 0$ and $(W'_4)^i = 1$ so that $W_4 + \Sigma_0(a_3)$ and $W'_4 + \Sigma_0(a'_3)$ are equal at the i^{th} bit position; $i = 9, 18$ and 29.
2. Similarly, $(W_4)^i$ and $(W'_4)^i$ have difference opposite to the difference in $(\Sigma_1(e_3))^i$ and $(\Sigma_1(e'_3))^i$ at bit positions $i = 6, 20$ and 25.

All the 6 bit differences will be cancelled if the conditions shown in Table 2, Step 4, column a_k are met. Note that this is not a necessary way of cancelling the differences, other possibilities exist when the sum of the terms in a_4 and a'_4 may behave as desired. In particular, we do not use bit carries in addition modulo 2^{32} to cancel these type of differences like Wang et. al do for SHA-1 [21]. We use XOR differences only, unlike [21] where modular differences are used.

Conditions due to register e_4 : Having cancelled the 6 bit differences to obtain $\Delta(a_4) = 0$, it can be seen that 3 bits from $\Delta(W_4)$ will certainly propagate into $\Delta(e_4)$ because there is no Σ_0 term in calculating e_4 and e'_4 . If the differential path is to be followed, then these 3 differing bits in W_4 and W'_4 should not carry forward to other positions. Carry propagation to other bits will cause problems in adjusting the register differences in next steps since any single bit difference in a or e register is expanded into 3 bit differences by the operators Σ_0 and Σ_1 . We have chosen the word differences in next steps considering these positions by following the linear (XOR) characteristics. It is possible to allow some bit carries here but it seems that it will only reduce the probability of the differential path.

To complete the analysis of step 4, we finally look at the difference $\Delta(e_4)$. The registers e_4 and e'_4 are computed as follows:

$$\begin{aligned}
 e_4 &= d_3 + \Sigma_1(e_3) + f_{IF}(e_3, f_3, g_3) + h_3 + K_4 + W_4, \\
 \text{and } e'_4 &= d'_3 + \Sigma_1(e'_3) + f_{IF}(e'_3, f'_3, g'_3) + h'_3 + K_4 + W'_4.
 \end{aligned}$$

In these two computations, bits 6, 20 and 25 corresponding to Σ_1 rotations of the differing bit 31 in e_3 have already been taken care of while considering a_4 . Bit numbers 9, 18 and 29 are the places where W_4 and W'_4 differ and these differences are required to be propagated to Δe_4 . Since $d_3 = d'_3$, $h_3 = h'_3$ and $f_{IF}(e_3, f_3, g_3) = f_{IF}(e'_3, f'_3, g'_3)$;

$$\text{if we write } \quad rest = \Sigma_1(e_3) + f_{IF}(e_3, f_3, g_3) + h_4 + K_4,$$

$$\begin{aligned} \text{then} & \quad e_4 = rest + W_4, \\ \text{and} & \quad e'_4 = rest + W'_4. \end{aligned}$$

If the i^{th} bit of $rest$ is 0 and there is no carry into the i^{th} bit while addition with W_4 takes place, then the XOR difference $W_4 \oplus W'_4$ will propagate into $e_4 \oplus e'_4$ as desired. Alternately, if the i^{th} bit of $rest$ is 1 and there is a carry into the i^{th} bit while addition with W_4 takes place, then too the XOR difference $W_4 \oplus W'_4$ will propagate into $e_4 \oplus e'_4$.

Thus either we would like no carry propagation in e_4 and e'_4 at bits 6, 20 and 25 if $rest$ is 0 at these bit positions or we would like carry propagation in both these registers if $rest$ is 1 at these bits. We do not have a deterministic way to ensure this since we do not have complete freedom to choose the registers and the message words as desired at this stage. However, the probability of the carries to happen as desired can be increased if we set other free bits of W_4 and W'_4 according to the following conditions :

1. if $rest^9$ is 0 then $W_4^7 = W_4^8 = 0$.
2. if $rest^9$ is 1 then $W_4^7 = W_4^8 = 1$.
3. if $rest^{18}$ is 0 then $W_4^{10} = W_4^{11} = \dots = W_4^{17} = 0$.
4. if $rest^{18}$ is 1 then $W_4^{10} = W_4^{11} = \dots = W_4^{17} = 1$.
5. if $rest^{29}$ is 0 then $W_4^{26} = W_4^{27} = W_4^{28} = 0$.
6. if $rest^{29}$ is 1 then $W_4^{26} = W_4^{27} = W_4^{28} = 1$.

In setting these conditions, we have used the bits between 6, 9, 20 and 9, 18 and 29 which are not restricted.

Similarly we have set conditions for other steps so that the messages follow the differential path as desired.

6.2 Method to Satisfy Conditions in Table 2

First 4 words in the differential path are free and hence we choose them randomly. Thereafter, many conditions in Table 2 are easy to fulfill as they depend only on word W_k in step k . Some of the conditions on registers can be tackled by suitably choosing the word W_k at that step which we can choose as desired. However, there may be instances when a previously selected message word causes impossible condition at a later step. As an example, we may not get the bit carry conditions for register e_4 as described previously. Also we wish to have e_6^{31} following a particular pattern at step 8 whereas this bit has been set at step 6 itself. In such contradicting cases, we choose another message word randomly at the previous step where the condition was breaking down. Then we apply message modification techniques from that step onwards and continue the search process for further steps. We search incrementally proceeding further only when all the conditions at a step are fulfilled and the differential path is as desired. The differential path in Table 1 holds with probability 2^{-49} , but with the procedure described above, we are able to get a much higher probability. In fact, Steps 0 to 7 become very easy to fulfill with the message modification and we are able to satisfy all the conditions till Step 7 in about a minute on an ordinary

PC. The only difficult conditions are those imposed due to a_8 . We could find a colliding message pair following exact differential characteristic in time varying from about 40 minutes to a couple of hours on an ordinary PC. Repeatedly running the program we could generate many such pairs. We show two such colliding pairs of messages.

6.3 Colliding Message Pairs for 18-Step SHA-256

Tables 3 and 4 show the message pairs found using the techniques described previously. All 18 words of the messages are given in the tables. First 16 words can be used to compute the last two words using the message expansion of SHA-256. Similar method can be used for finding 9-round pseudo collisions for SHA-256 as well. Since we can already find message pairs colliding for 18-step SHA-256 with the standard IV, the only utility for such an exercise would be to see how easy it becomes to find these pseudo collisions due to the benefits of relaxing the IV conditions. However, we found that the time required to find a 9-round pseudo collision is only marginally less than the time required to find an 18-step collision. An example of such a pseudo collision is provided in 17.

Table 3. Colliding message pair for 18 step SHA-256 with standard IV. These two messages follow the differential path given in Table 1

M ₁	0-7	ccea5c17	53ad1a2d	141db23c	b6acfaa8	5ee7fe4d	53c5b764	2bf20d44	87d63bf6
	8-15	63a07869	f305fdea	26ee271f	b973b91c	d0f87828	b724a487	a295fa2a	0a67c97a
M ₂	0-7	ccea5c17	53ad1a2d	141db23c	36acfaa8	7cf3fc0d	1140a7fc	09e60f04	87d63bf6
	8-15	41b47a29	f305fdea	26ee271f	3973b91c	d0f87828	b724a487	a295fa2a	0a67c97a

Table 4. Another colliding message pair for 18 step SHA-256 with standard IV. These two messages also follow the differential path given in Table 1

M ₁	0-7	e4919421	aa75e4fe	8548d0e0	9c1888f7	1da3fc3d	a11f7a02	bb463b64	e9b2836f
	8-15	323accf28	8097e497	4343b78b	dc484e91	bf5885b4b	8401140a	42499da1	f88a3e2e
M ₂	0-7	e4919421	aa75e4fe	8548d0e0	1c1888f7	3fb7fe7d	e39a6a9a	99523924	e9b2836f
	8-15	102acd68	8097e497	4343b78b	5c484e91	bf5885b4b	8401140a	42499da1	f88a3e2e

It seems possible to use neutral bits to increase the efficiency of the search for finding message pairs following the given differential path. We experimented with this idea and found that the gains are not significant. More details about our experiments with neutral bits are available in 17.

7 Using Coding Theoretic Methods to Find Linear Differential Paths for Reduced Round SHA-256

In 15 and 14 coding theoretic techniques were used to search for differential paths in SHA-1. Extension to SHA-2 was mentioned in 11. We describe a new way of forming parity check equations and then find low weight codewords for the corresponding generator matrix. Each of these codewords can be used to build a differential characteristic for reduced round SHA-256. This method results in tackling up to 23-step reduced SHA-256.

7.1 A New Way of Constructing Parity Check Equations

Tackling message expansion in SHA-2 can be a problem. A non-zero value of ΔW_i for $i \geq 16$ necessitates tackling the recursion for message expansion. So one way to avoid this is to ensure that $\Delta W_i = 0$ for $i \geq 16$. Clearly, this cannot work for full SHA-2. But, for reduced round versions, one can find differential paths using this approach, as we describe below.

The technique described below assumes a local collision \mathcal{L} . The description is not for any particular local collision. It holds for any local collision. Obtaining a particular local collision requires certain linear approximations of the constituents of the SHA-256 round function. This converts the round function into a linear map based on which we define our linear code. We note that the linear code is not straightforwardly obtained from the linear map.

A message consists of 16 32-bit words for a total of 512 bits. We use the Chabaud-Joux [3] type disturbance vector approach. Let $DV = \{d_0, d_1, d_2, \dots, d_{255}\}$ be a 256-bit disturbance vector. If $d_i = 1$ then the two initial messages differ in their i^{th} bit, and further message bits differ as per the local collision.

We do not consider a 512-bit DV for the following reason. A local collision defines the differences of 9 words of messages and only the first 16 words of SHA-256 are unrestricted. Thereafter the message words are calculated using the message expansion recurrence. This implies that a local collision can not be started after first 8 steps without affecting the message expansion.

Let us now describe the linear code that we require. This is done in two steps. In the first step, we express ΔW_i ($i \geq 16$) in terms of d_0, \dots, d_{255} . In the second step, we define the parity check equations for the code by setting $\Delta W_i = 0$ for $i \geq 16$. Thus, any DV (d_0, \dots, d_{255}) which satisfies these parity check equations is a codeword. Our task then is to look for a low weight codeword as this gives a differential path with a small number of local collisions.

It is clear that such codes can be formed as long as there are less than 256 parity check equations. If we apply this procedure up to N rounds (corresponding to step $N - 1$), then we will obtain $32(N - 16)$ parity check equations. Thus, the maximum N that we can use with this method is $N = 23$. The minimum value of N is clearly 17. Since we already report 18-round collision, we do not consider $N = 17$ and 18. Instead we report differential paths from 19 to 23 rounds.

The first task is to express ΔW_i ($i \geq 16$) in terms of d_0, \dots, d_{255} . We describe how this is done. For any local collision \mathcal{L} , the first word determines the next eight words. Consider the 32-bit vector $(d_0, 0, \dots, 0)$, where d_0 is treated as a (binary) variable. Then \mathcal{L} defines the next 8 32-bit words. At this point, the first 9 words have been defined. The rest 7 are taken to be zero. For $i \geq 16$, ΔW_i is now obtained using the message recursion. This expresses all the ΔW_i s ($i \geq 16$) as linear function of d_0 . Next consider the 32-bit vector $(0, d_1, 0, \dots, 0)$; use \mathcal{L} to obtain the next eight words and the message expansion recursion to express ΔW_i s ($i \geq 16$) as linear function of d_1 . Now, for the 32-bit vector $(d_0, d_1, 0, \dots, 0)$, we can express ΔW_i s ($i \geq 16$) as linear function of d_0 and d_1 by XORing the separate linear functions corresponding to d_0 and d_1 . Clearly, the procedure can be extended to the entire DV (d_0, \dots, d_{255}) . The exact details are given in Table 5.

Table 5. Algorithm for generating parity check equations for linearized N step SHA-256

```

external LC( $x$ ) : accepts a 32 bit input  $x$  and returns 9 words of 32 bits conforming to the local collision chosen.


---


Set  $\Delta W_{final} := (U_0, U_1, \dots, U_{N-1}) \quad U_i \in \{0, 1\}^{32}$ 
Set  $\delta W_{cur} := (V_0, V_1, \dots, V_8) \quad V_i \in \{0, 1\}^{32}$ 
Initialize  $\Delta W_{final}$  and  $\delta W_{cur}$  to all zeros.

For( $i = 0$  to 8){
  For( $j = 0$  to 31){
    set  $D := (0, 0, \dots, d_{32i+j}, 0, \dots, 0);$  /* The  $j^{th}$  bit of  $D$  is given by  $d_{32i+j}$ .
      Each  $d_n \in \{0, 1\}$  is the component of the disturbance vector and  $D \in \{0, 1\}^{32}$  */
    set  $\delta W_{cur} := LC(D);$ 
    For( $k = i$  to  $i + 8$ ){
       $\Delta W_{final}[k] = \Delta W_{final}[k] \oplus \delta W_{cur}[k - i];$ 
    }
  }
}
/* At this point the  $\Delta W_{final}$  list contains  $W_i \oplus W_i'$  for  $0 \leq i < 16$  */

Obtain  $\Delta W_i$  for  $16 \leq i < N$  using linearized message expansion of SHA-256.
Equate all 32 bits of  $\Delta W_i$  for  $i \geq 16$  to zero to get  $32 * (N - 16)$  parity check equations.


---



```

Methods presented in [2], [8] and [19] are used to search for low weight codewords from the check-matrices (and the corresponding generator matrices) obtained using the algorithm in Table 5. Codewords of least weight found and the linear differential path for that codeword are shown in Section 8.

8 Results and Comparison to Previous Work

Low weight disturbance vectors are searched for reduced round SHA-256 by using the probabilistic methods described in [2], [8] and [19]. The minimum weights of codewords found are listed in Table 6. For 19-step SHA-256 the weight of the codeword found is 15 for both GH and SS_5 local collision. This means that 15 local collisions are interleaved to obtain the 19-step characteristic. Interestingly, all the 15 local collisions start at the same word for both GH and SS_5 . Thus the case of 19-step characteristic can be considered as consisting of a single local collision starting at Step 3 where the initial message difference is a word with weight 15 bits. There is no colliding differential path known before this work using the linearized local collision. Using this technique, the best known 19-step differential path is for a near collision consisting of 23 GH local collisions [11]. As has already been noted in [11], the GH local collision causes certain impossible conditions in the search for actual colliding pairs. The use of SS_5 local collision ensures that we do not face two types of impossible conditions.

For 20 to 23 steps, no differential path using a linearized local collision is known so far. We provide the first differential paths for these cases using the linearization technique. For 23-step SHA-256, the size of the corresponding generator matrix is 32×256 , i.e. there are only 32 codewords of length 256. It is possible to do exhaustive search on this size, hence we did not use the probabilistic methods for this case. For the 23-step case, the reported codeword weight is actually the best possible. All these differential paths are reported in [17].

Table 6. Summary of results. Least weight of the codeword found using different local collisions. For 23 step case, the codeword weight is obtained by exhaustive search. For all other cases, methods described in [2], [8] and [19] are used.

Step i	Size of Check matrix	using GH	using SS ₅
18	-	1	1
19	96×256	15	15
20	128×256	33	31
21	160×256	45	45
22	192×256	59	60
23	224×256	79	75

Acknowledgements

We would like to thank Christian Rechberger for carefully reading an earlier version of this paper and giving helpful suggestions. He also mentioned that analysis similar to ours appears in [9].

References

1. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
2. Canteaut, A., Chabaud, F.: A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece’s Cryptosystem and to Narrow-Sense BCH Codes of Length 511. IEEE Transactions on Information Theory 44(1), 367–378 (1998)
3. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
4. Cramer, R.J.F. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
5. Gilbert, H., Handschuh, H.: Security Analysis of SHA-256 and Sisters. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 175–193. Springer, Heidelberg (2004)
6. P. Hawkes, M. Paddon, G.G. Rose. On Corrective Patterns for the SHA-2 Family. Cryptology eprint Archive (August 2004), <http://eprint.iacr.org/2004/207>
7. Hölbl, M., Rechberger, C., Welzer, T.: Finding Message Pairs Conforming to Simple SHA-256 Characteristics. In: Lucks, S., Sadeghi, A.-R., Wolf, C. (eds.) Stefan Lucks, Ahmad-Reza Sadeghi, and Christopher Wolf, editors, Preliminary Proceeding Records of WEWoRC 2007 - Western European Workshop on Research in Cryptology, Bochum, Germany, pp. 21–25 (2007), <http://www.hgi.rub.de/weworc07/PreliminaryConferenceRecord.pdf>
8. Leon, J.S.: A Probabilistic Algorithm for Computing Minimum Weights of Large Error-Correcting Codes. IEEE Transactions on Information Theory 34(5), 1354–1359 (1988)
9. Matusiewicz, K.: Analysis of Modern Dedicated Cryptographic Hash Functions. PhD thesis, Macquarie University (August 2007), <http://www.ics.mq.edu.au/~kmatus/Matusiewicz-PhDthesis.pdf>

10. Matusiewicz, K., Pieprzyk, J., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of simplified variants of SHA-256. In: Wolf, C., Lucks, S., Yau, P.-W. (eds.) WEWoRC 2005 - Western European Workshop on Research in Cryptology, Leuven, Belgium, July 5-7, 2005. LNI, vol. 74, pp. 123–134. GI (2005)
11. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of Step-Reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
12. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of Step-Reduced SHA-256. Cryptology eprint Archive (March 2008), <http://eprint.iacr.org/2008/130>
13. Nikolić, I., Biryukov, A.: Collisions for Step-Reduced SHA-256. In: Nyberg, K. (ed.) Fast Software Encryption 2008. volume Pre-proceedings version of Lecture Notes in Computer Science, pp. 1–16. Springer, Heidelberg (2008)
14. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 78–95. Springer, Heidelberg (2005)
15. Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)
16. Sanadhya, S.K., Sarkar, P.: New Local Collisions for the SHA-2 Hash Family. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 193–205. Springer, Heidelberg (2007)
17. Sanadhya, S.K., Sarkar, P.: Attacking Reduced Round SHA-256. Cryptology eprint Archive (March 2008), <http://eprint.iacr.org/2008/142>
18. Secure Hash Standard. Federal Information Processing Standard Publication 180-2. U.S. Department of Commerce, National Institute of Standards and Technology (NIST) (2002), <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>
19. Stern, J.: A Method for Finding Codewords of Small Weight. In: Wolfmann, J., Cohen, G. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989)
20. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer [4], pp. 1–18
21. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
22. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer [4], pp. 19–35
23. Yoshida, H., Biryukov, A.: Analysis of a SHA-256 Variant. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 245–260. Springer, Heidelberg (2006)

DAKOTA – Hashing from a Combination of Modular Arithmetic and Symmetric Cryptography

Ivan B. Damgård¹, Lars R. Knudsen², and Søren S. Thomsen²

¹ Department of Computer Science, University of Aarhus,
IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark

² Department of Mathematics, Technical University of Denmark,
Matematiktorvet 303S, DK-2800 Kgs. Lyngby, Denmark

Abstract. In this paper a cryptographic hash function is proposed, where collision resistance is based upon an assumption that involves squaring modulo an RSA modulus in combination with a one-way function that does not compress its input, and may therefore be constructed from standard techniques and assumptions. We are not able to reduce collision finding to factoring, but on the other hand, our hash function is more efficient than any known construction that makes use of modular squaring.

1 Introduction

With the emergence of a large number of attacks (e.g. [1, 2, 4, 11, 27, 28, 29]) on many dedicated hash functions, the development of alternatives based on different design principles has become increasingly desirable. While much research has shifted towards hash functions based on block ciphers, some recent proposals such as [5, 7, 12, 16, 17] aim for some sort of “provable security”. Provable security is in quotation marks because security proofs are always based on some unproven assumption.

The downside to provably secure hash functions is that they are often much less efficient than commonly used alternatives like MD5 [24] and SHA-1 [21], and with the arguably most common application of hash functions being the pre-processing of a message in a digital signature scheme with the intention of making the entire signing process faster, a slow hash function loses some of its justification. Hence, speed cannot be overlooked altogether, but keeping in mind the large number of recent attacks on fast dedicated hash functions, it may be beneficial to reconsider the scaling of speed vs. security.

In this paper, we propose a reasonably fast hash function based partly on number theoretic principles. We prove that finding collisions is as hard as breaking a computational assumption that involves squaring modulo an RSA modulus in combination with a one-way, collision resistant function that *does not need to compress* (hence it can be injective). We propose different variants of such a function, and we urge the community to assist in analysing these variants and to possibly propose others.

2 Hash Function Security

Three types of attack on hash functions are usually mentioned in the literature. These are the preimage attack, the second preimage attack and the collision attack, with expected complexities for an ideal m -bit hash function of, respectively, 2^m , 2^m , and $2^{m/2}$. In most cases these are also the conjectured complexities of a newly designed hash function, and if an attack of lower complexity is discovered, then the hash function is usually considered broken.

Finding second preimages in a hash function is at least as hard as finding collisions, since given a second preimage, one immediately has a collision. Moreover, if we let H be a hash function that accepts inputs which are (much) longer than the outputs of H , then an algorithm that finds preimages might be given the hash value $H(x)$ for some x . With good probability, the algorithm will return a preimage \tilde{x} of $H(x)$ such that $x \neq \tilde{x}$. This also yields a collision. Hence, a collision resistant hash function may reasonably be considered resistant also to preimage attacks.

Hash functions aimed towards the use in applications such as digital signatures, and to some extent commitment schemes and data integrity, must be collision resistant. Hence, the complexity of finding collisions must be high enough that the task is infeasible, preferably for many years to come. For such hash functions it makes sense for the designers to indicate one claimed attack complexity, a complexity that is a lower bound for all three mentioned types of attack.

This is the choice we have made here. We shall not claim that our hash function behaves as a random oracle. We simply claim that it is collision intractable.

3 The Proposal

The hash function proposal of this paper is now presented. We call this hash function DAKOTA.

The basic idea can be seen as a further development of earlier hash functions whose security is provably reducible to factoring [14,9]. A representative example of these ideas is where the compression function h maps as $h : \{0, 1\} \times SQ(n) \rightarrow SQ(n)$, and is defined by $h(b, y) = a_b y^2 \bmod n$, where a_0, a_1 are randomly chosen squares modulo RSA modulus n , and $SQ(n)$ is the set of squares modulo n . It is easy to show that an algorithm that finds collisions for h can be used to factor n with probability $1/2$. The actual hash function H is obtained by iterating h in a standard Merkle-Damgård construction, using a random square y_0 as initial value.

An alternative formulation, better suited for generalisation is to define a function $f : \{0, 1\} \rightarrow SQ(n)$, where $f(b) = a_b$ and write $h(b, y) = f(b)y^2 \bmod n$. The assumption that h is collision intractable can be phrased as saying that it is hard to find distinct inputs $(b, y), (b', y')$ such that $f(b)f(b')^{-1} = (y'y^{-1})^2 \bmod n$.

While this construction is very inefficient, since we spend a modular squaring to hash a single bit, efficiency can be improved by extending the input domain of f : we can define instead f to take input from $\{0, 1\}^t$, and select 2^t random

squares as output values. This is t times faster than the basic idea. The reduction to factoring can be constructed to have success probability independent of t , see [9]. However, this idea is of course only practical for small values of t .

A natural alternative seems to be to define an efficient algorithm for computing f , instead of specifying it as a table with all output values. In this way we can allow f to take large strings as input and gain efficiency. The price to pay, as we shall see, is that we can no longer prove that security is equivalent to factoring – we return later to a discussion of what we can prove instead. A more immediate practical problem, however, is that the security of the original construction requires f to map into $SQ(n)$, and it is not clear how we can ensure that this happens when we have too many inputs to be able to specify the function by a table. To hit $SQ(n)$, it seems that the algorithm computing f would have to either (a) square a known value or (b) make use of the factorisation of n . In the case of (a), this would mean that the adversary would know the square root of the output of f , and this would invalidate the reduction to factoring. With respect to (b), this does not work because f has to be a public function, so it cannot be based on the secret factors of n .

We therefore propose below a variant of the idea, that permits f to map into all of \mathbb{Z}_n while still allowing a security proof.

To this end, we assume we are given a probabilistic algorithm \mathcal{G} that on input a security parameter k produces an RSA modulus $n = pq > 2^k$, where $p \equiv q \equiv 3 \pmod{4}$ (this means that -1 is a quadratic non-residue mod n , and that squaring mod n is a bijection on the set of quadratic residues, see e.g. [18]), together with a description of a function $f : \{0, 1\}^k \rightarrow \mathbb{Z}_n$. As we shall see, f will have to be one-way and collision intractable, but since f does not compress its input, there is no circularity here. Finally, it chooses $r \in \mathbb{Z}_n^*$ at random, sets $s = r^2 \pmod{n}$ and returns (f, n, s) (p, q , and r are discarded in a secure manner).

Using the output of \mathcal{G} , we define a compression function $h : \{0, 1\}^k \times \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ as follows:

$$h(x, y) = (f(x)y)^2 \pmod{n}.$$

From this compression function we can build a collision intractable hash function under the following assumption.

Assumption 1. Consider a probabilistic polynomial time algorithm that takes as input f, n as produced by \mathcal{G} on input k and outputs x, \tilde{x}, z . Then the probability that $x \neq \tilde{x}$ and $f(x)/f(\tilde{x}) = \pm z^2 \pmod{n}$ is negligible.

Assumption [1] leads to some requirements on the function f . For a discussion, see Section [3.1].

We now define our hash function H by a standard Merkle-Damgård construction [10, 19]: split the input message x into blocks of length k bits, call them x_1, \dots, x_t . We assume for simplicity that x has been padded to a length divisible by k . Then define $y_0 = s$, and $y_i = h(x_i, y_{i-1})$ for $1 \leq i \leq t$. Finally, set $H(x) = y_t$.

Collision intractability of H does not follow as usual with this type of iterated construction because our compression function is not collision intractable (inputs of the form $(x, y), (x, -y)$ collide). However, we can still prove:

Theorem 1. *The hash function H as described above is collision intractable under Assumption [1](#).*

Proof. We assume we are given an algorithm \mathcal{A} that finds collisions with probability ϵ . We then build an algorithm that breaks Assumption [1](#): we get f, n as input, we choose r at random in \mathbb{Z}_n^* , set $s = r^2 \bmod n$ and we give f, n, s to \mathcal{A} .

Assume now that a collision has been found by \mathcal{A} , for different inputs x_1, \dots, x_t and $\tilde{x}_1, \dots, \tilde{x}_{\tilde{t}}$. This gives rise to two sequences of values y_0, \dots, y_t and $\tilde{y}_0, \dots, \tilde{y}_{\tilde{t}}$, where $y_t = \tilde{y}_{\tilde{t}}$. This implies that we have $h(x_t, y_{t-1}) = h(\tilde{x}_{\tilde{t}}, \tilde{y}_{\tilde{t}-1})$, or equivalently

$$(f(x_t)/f(\tilde{x}_{\tilde{t}}))^2 = (\tilde{y}_{\tilde{t}-1}/y_{t-1})^2 \bmod n.$$

If $f(x_t)/f(\tilde{x}_{\tilde{t}}) \neq \pm \tilde{y}_{\tilde{t}-1}/y_{t-1} \bmod n$, then we can efficiently factor n by computing (e.g.) $p = \gcd(f(x_t)/f(\tilde{x}_{\tilde{t}}) - \tilde{y}_{\tilde{t}-1}/y_{t-1}, n)$. Factoring n in particular allows breaking Assumption [1](#).

So we may assume that $f(x_t)/f(\tilde{x}_{\tilde{t}}) = \pm \tilde{y}_{\tilde{t}-1}/y_{t-1} \bmod n$. Now, if $x_t \neq \tilde{x}_{\tilde{t}}$, we can break Assumption [1](#), using the fact that we can compute a square root z of $\tilde{y}_{\tilde{t}-1}/y_{t-1} \bmod n$. This is because we already know a square root of y_{t-1} , namely $f(x_{t-1})y_{t-2} \bmod n$ (if $t > 1$) or r (if $t = 1$); and by the same argument we also know a square root of $\tilde{y}_{\tilde{t}-1}$.

On the other hand, if $x_t = \tilde{x}_{\tilde{t}}$ then $1 = f(x_t)/f(\tilde{x}_{\tilde{t}}) = \pm \tilde{y}_{\tilde{t}-1}/y_{t-1} \bmod n$. Note that $\tilde{y}_{\tilde{t}-1}/y_{t-1} \in SQ(n)$ since all y -values are squares modulo n , but $-1 \notin SQ(n)$, so we conclude that $\tilde{y}_{\tilde{t}-1}/y_{t-1} = 1 \iff \tilde{y}_{\tilde{t}-1} = y_{t-1}$, and we can now repeat the same argument.

The only way in which this can fail to produce a contradiction with Assumption [1](#) is if we end up concluding that $y_t = \tilde{y}_{\tilde{t}}, y_{t-1} = \tilde{y}_{\tilde{t}-1}, \dots, y_0 = \tilde{y}_{\tilde{t}-t}$, where we assume without loss of generality that $\tilde{t} \geq t$. Since the two inputs are different it must be that $\tilde{t} > t$, whence we have that

$$s = y_0 = h(\tilde{x}_{\tilde{t}-t}, \tilde{y}_{\tilde{t}-t-1}) = (f(\tilde{x}_{\tilde{t}-t})\tilde{y}_{\tilde{t}-t-1})^2 \bmod n.$$

In other words, we have produced a square root of s . Since r was uniformly chosen, there is a probability of $1/2$ that the square root we find here is different from $\pm r \bmod n$, in which case we can factor n and break Assumption [1](#). Hence if \mathcal{A} finds a collision with probability ϵ , we can break the assumption with probability at least $\epsilon/2$. \square

Remark 1. If n can be factored, then Assumption [1](#) can be broken, and this may allow for an attack on the hash function itself. Hence, n must be generated in such a way that nobody knows its factorisation. Boneh and Franklin [3](#) have described efficient techniques to do this securely.

3.1 Notes on Assumption [1](#)

If f can be inverted on a non-negligible subset of \mathbb{Z}_n , then Assumption [1](#) can be broken. One simply chooses \tilde{x} and z , and computes $f^{-1}(z^2 f(\tilde{x}))$. In particular, it must be infeasible to find a zero of f , since otherwise Assumption [1](#) is broken with $z = 0$ and any \tilde{x} .

On the other hand, the image of f may only be a small subset of \mathbb{Z}_n , and in such cases the ability to find a preimage of a given image of f may not be a problem. This is due to the fact that the size of the image of the function $F(x, \tilde{x}) = f(x)/f(\tilde{x}) \bmod n$ can be made much smaller than n , for instance $n/2^{128}$. In this case, assuming that the outputs of F are uniformly distributed in \mathbb{Z}_n , the probability that for a given z , an input (x, \tilde{x}) to F exists such that $F(x, \tilde{x}) = \pm z^2 \bmod n$, is only about 2^{-128} . Of course, if f expands, then the efficiency of the hash function decreases.

It is clear why f must be collision intractable, as mentioned: if $f(x) = f(\tilde{x})$ for $x \neq \tilde{x}$, then Assumption [II](#) is broken with $z = 1$. However, this possibility is excluded if we design f to be injective.

If f is one-way, and n cannot be factorised, then attacks where one chooses two values of (x, \tilde{x}, z) and computes the third will fail. Computing z for given (x, \tilde{x}) requires computing a square root modulo n . Computing, say, x given (\tilde{x}, z) , requires inverting f .

Another generic attack is to compute $z^2 \bmod n$ for many random values of z , likewise many values of $f(x)/f(\tilde{x}) \bmod n$, and hope for a collision. This is just a standard meet-in-the-middle attack which has complexity about \sqrt{n} and hence has no practical significance.

Whether a more efficient method to find collisions than by factoring n is possible, depends of course on whether the design of f interacts in some unfortunate way with arithmetic modulo n . For instance, one may try to find x, \tilde{x} such that $f(x) = -f(\tilde{x}) \bmod n$, or in general x, \tilde{x} such that $f(x) = \pm a^2 f(\tilde{x}) \bmod n$ for some a of the adversary's choice. We return to this question below.

3.2 Output Transformation

For most applications it is recommended that the output of H is not used as the final hash, but is instead fed to an output transformation function $\Omega : \mathbb{Z}_n \rightarrow \{0, 1\}^m$, where m is chosen such that the complexity of factoring n is no less than $2^{m/2}$. The intention is to obtain an output size corresponding to the security level of the hash function. In addition, Ω might be used to obfuscate the algebraic structure of the output. Any algebraic structure could compromise the security of schemes in which the hash function is used, particularly schemes that are based on modular arithmetic such as signature schemes based on RSA [\[25, 26\]](#). A third purpose of an output transformation may be to improve the preimage resistance of the hash function.

In order to provably extend the collision resistance of H to $\Omega \circ H$, Ω would itself have to be collision intractable. But in practice this may not be necessary: even if it is easy to find collisions for Ω , these do not necessarily lead to collisions for $\Omega \circ H$. In fact, it may be sufficient that Ω mixes the bits of its input well such that all output bits depend on all input bits, and that it is regular meaning that all 2^m preimage sets are of roughly the same size. Unless the hash function is primarily used for short messages, Ω does not have to be terribly fast, since it is only used once.

A concrete example of how the output transformation could work is as follows: Let G be a finite group of prime order and let g_1, \dots, g_u be chosen randomly in G . We choose the two integers t and u such that $t < \log_2 |G|$ and $tu \geq \log_2(n)$. Let $b = H(x) = b_1 \parallel \dots \parallel b_u$ be the output from the main hash function, where the length of each b_i is t . Since $tu \geq \log_2(n)$ this may require that b_u be padded with zeros – a simple padding scheme is fine here, since all inputs to the output transformation will have the same length. We then define the intermediate output $\omega(b)$ as $g_1^{b_1} \dots g_u^{b_u}$, that is, a single element from G .

It is well known [6] (and straightforward to show) that finding collisions for this mapping is as hard as solving the discrete logarithm problem (DLP) in G . There are well known constructions of such groups based on elliptic curves, where the DLP can be reasonably assumed to be hard, and where the representation of a group element is 200-400 bits long. Note also that by choosing t small, for instance ≤ 8 , we may perform the exponentiations $g_i^{b_i}$ by table lookups.

We recommend that a final bijective function based on symmetric cryptography is used on $\omega(b)$ to produce the final output $\Omega(b)$, in order to prevent the adversary from exploiting the algebraic properties of exponentiation in G . This function could be designed based on a block cipher in CBC mode as described in more detail in Section 4.1.

4 Proposals for f

In this section we describe a number of possible instantiations of f , the function used in the compression function of DAKOTA which must satisfy Assumption 1.

As mentioned, for f to satisfy Assumption 1, it cannot be easily invertible, and it must also be collision resistant. Hence, it might be desirable that f be injective. It should take a k -bit input and return an element of \mathbb{Z}_n .

One-wayness and collision resistance are necessary, but not sufficient conditions. As an example, consider $f(x) = x^2 \bmod n$, where $x > n/2$. It is generally believed that for a secure modulus n , this function is one-way and collision resistant. However, this is clearly a bad proposal, since given z and \tilde{x} , one may choose $x = z\tilde{x}$ and thus break Assumption 1.

If, however, one assumes that f is, in some sense, independent of arithmetic modulo n , or put differently, does not interact badly with arithmetic mod n , then, intuitively, it seems that one-wayness and collision resistance are indeed sufficient properties. This idea is made more concrete in the proposals below.

In the following we assume a modulus of size about 1024 bits. A proposal for f should be evaluated on its security properties, and also on its scalability, i.e. how easily it can be adapted to a new modulus of a different size.

4.1 Combining Modular Arithmetic with Symmetric Encryption

It is quite straightforward to construct a one-to-one function that has “nothing” to do with modular arithmetic in the sense that we make it hard for an adversary to choose inputs for which the outputs satisfy some algebraic relation mod n : one can simply encrypt the input under a fixed key using a symmetric algorithm,

say, a block cipher in CBC mode. To make life harder for the adversary, it may be desirable that all output bits depend on all input bits. This can be ensured by encrypting twice, reversing the order of the blocks for the second encryption. Unfortunately, anything based on encryption under a fixed key is easy to invert, and so this is not sufficient to get what we want.

On the other hand, if we are willing to use modular arithmetic, we can easily satisfy a different subset of our demands, namely we can build a fast one-way and one-to-one function by using modular squaring. Actually, for an RSA modulus, squaring is a four-to-one mapping, but if we constrain the input to be less than half the modulus, one can only find collisions by factoring the modulus, so this is “as good as” being one-to-one.

An obvious idea to get all the properties we want is now to combine the two ideas: first square to get the one-way property and then do AES encryption to obfuscate the algebraic structure. This results in the following proposal for f (keep in mind that k is the size (number of bits) of the input to f):

Proposal 1. We assume the size of n is 1025 bits, and we choose another RSA modulus n' of size 1024 bits. We let $k = 1022$ and x be the input, and let E_κ be AES encryption in CBC mode with key κ . κ_1 and κ_2 are two fixed AES keys.

- Let $u = x^2 \bmod n'$.
- Let $v = E_{\kappa_1}(u) = v_1 \| \cdots \| v_8$.
- Let $f(x) = E_{\kappa_2}(v_8 \| \cdots \| v_1)$ (notice that the order of the blocks of v is reversed).

The specific choice of input and output sizes is just to make sure that inputs are less than $n'/2$ and that outputs are less than n .

With this construction of f , we can hash about 128 bytes using two modular squarings, a multiplication and AES encryption of 256 bytes.

How hard is it for the adversary to break our assumption for this choice of f ? The good news is that the straightforward ways to attack will not work: the adversary may compute $f(x), f(\tilde{x})$ and try to extract a square root of $\pm f(x)/f(\tilde{x}) \bmod n$, he can try to invert f or he can try to find a collision of f . All three types of attack are as hard as factoring n or n' .

However, for attacks that choose a and then try to find x, \tilde{x} such that $f(x)/f(\tilde{x}) = \pm a^2 \bmod n$, we can only base ourselves on the heuristic that the design of f is sufficiently incompatible with arithmetic modulo n for this to be infeasible.

Proposal 1 is easily adapted to an RSA modulus n of a different size: simply choose n' as an RSA modulus smaller than $n/2$, and ensure $0 \leq x < n'/2$. Furthermore, the double CBC encryption will have to accommodate a different number of blocks.

4.2 A Proposal Using a k -Bit Permutation

If a (suitable) k -bit permutation g is available, then one might define f simply as $g(x) \oplus x$. This is a standard method of obtaining a one-way function from

a (“good”) invertible permutation, see e.g. [19]. For $k = 1024$ we suggest the following definition of f .

Proposal 2. Define f as $g(x) \oplus x$, where g is defined as follows. Let τ be an invertible function that transforms the 1024-bit string x into an 8×8 matrix A of 16-bit values. Let A_i be row i of A , and let \mathbf{E} be the function operating on A by replacing A_i with $E_i(A_i)$, $0 \leq i < 8$, where E_κ is the AES encryption function with key κ . Define a *round* as

$$A \leftarrow \mathbf{E}(A)^T$$

(where $(\cdot)^T$ is the transpose operator). Perform 4 such rounds. Finally, let $g(x) = \tau^{-1}(A)$.

Every state bit depends on every input bit after just two rounds. It seems very hard to identify a useful structure in g or its inverse.

Proposal 2 is easily adapted to, e.g., a 2049-bit modulus as follows: Let x be a 2048-bit input string, and form from x a 16×16 matrix of (8-bit) bytes. Define \mathbf{E} as above, only with $0 \leq i < 16$, and perform again 4 rounds as defined above.

5 Performance

Our proposal may be compared to the basic version of VSH [7] (see also Section 6) as follows: in both hash functions a multiplication and a squaring modulo n must be performed for each message block, plus an overhead which in the case of VSH is due to the computation of the product of small primes, and in our case is due to the evaluation of f . Our hash function is likely to perform better for one important reason: the size of a message block in our case is up to $\log_2(n)$ bits, whereas in the case of (basic) VSH it is the largest number t such that the product of the first t primes is less than n (in [7], t is estimated to be approximately $\frac{\log n}{\log \log n}$). As an example, with n begin a 1024-bit modulus, the size of a message block in our hash function may be up to 1024 bits, and in VSH it would be 131 bits.

There may also be an important difference in efficiency between evaluating f and computing the product of up to t primes.

There are faster versions of VSH that use larger message blocks and precompute some products of the small primes. These versions require a larger modulus to be used, and reliably comparing fast VSH with DAKOTA requires implementations using similar optimisations, compilers, processors etc. According to measurements presented in [7], fast VSH with claimed security equivalent to factoring a 1024-bit modulus reaches speeds of around 840 cycles/byte (on a 1GHz Pentium III, which is a 32-bit processor).

The two described versions of DAKOTA have been implemented with random choices of the moduli n (1025 bits) and n' (1024 bits), the square s and AES keys κ_1 and κ_2 . We used our own C implementation of AES, which does not achieve quite the same speeds as, e.g., those of the Crypto++ [8] library.

Table 1. Speed comparison of DAKOTA with SHA-256 (see text for details)

Hash function	Approximate speed (cycles/byte)	
	32-bit	64-bit
SHA-256	20	20
DAKOTA (Proposal 1)	385	170
DAKOTA (Proposal 2)	330	170

For large integer arithmetic, the GMP (GNU Multiple Precision) arithmetic library [\[13\]](#) has been used (version 4.2.2). The implementation was compiled and run in 32-bit and 64-bit modes, see Table [1](#) for benchmarks. In the table are also included benchmarks for the Crypto++ [\[8\]](#) (version 5.5) implementation of SHA-256 [\[22\]](#). All benchmarks refer to a test on a 2.4GHz Intel Core 2 Duo processor. The DAKOTA implementations were compiled using gcc version 4.1.3 (with optimisation flags `-static -O2 -fomit-frame-pointer`) in GNU/Linux Ubuntu 7.10. Further optimisations of the implementations are almost certainly possible.

We have not performed tests using larger modulus sizes, although in order to achieve security (with respect to collisions) comparable to, e.g., SHA-256, a modulus size of about 3072 bits would be needed, according to estimates by NIST [\[23\]](#).

5.1 Montgomery Multiplication

The Montgomery reduction [\[20\]](#) is a means to speed up modular computations. Let R be an integer, which in our case could be chosen to some power of two. In a modular multiplication modulo n of integers x and y , one first computes $x' = xR \bmod n$ and $y' = yR \bmod n$. The Montgomery reduction of $x'y'$ is then $x'y'R^{-1} = xyR \bmod n$.

In our proposal we are computing a series of values of the form $y_{i+1} = (f(x_i)y_i)^2 \bmod n$. We will introduce a variant of the Montgomery reduction tailored for our computations.

Let $\tilde{y}_0 = y_0R^3 \bmod n$, then compute the Montgomery reduction of $f(x)\tilde{y}_0$ which yields $w = f(x)y_0R^2 \bmod n$, then compute the Montgomery reduction of w , which is

$$\tilde{y}_1 = (f(x)y_0)^2R^3 \bmod n = y_1R^3 \bmod n.$$

This method can be iterated such that given $y_iR^3 \bmod n$ we get $y_{i+1}R^3 \bmod n$ using one multiplication, one squaring and two Montgomery reductions. When all message blocks have been processed in the hash function, the constant R^3 can be removed.

6 Related Work

A number of hash functions claiming to obtain provable security have been proposed in the past. Some examples are now mentioned.

Goldwasser-Micali-Rivest. Following ideas of [14], a provably collision intractable hash function can be constructed as described in Section 3 (see also [9]). For completeness, we describe the construction again here: Let a_0, a_1 be random squares modulo RSA modulus n . All values are public, but the factors of n are secret. Define the compression function $h : \{0, 1\} \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$ by

$$h(x, y) = a_x y^2 \bmod n.$$

It follows that a collision gives a square root modulo n which (with probability $1/2$) can be used to factor n . Variants that are t times faster have been proposed [9]; these use 2^t public squares and preserve the collision intractability.

VSH. A more recent example of a provably collision resistant hash function is VSH [7], which is roughly defined as follows. Let n be a public RSA modulus. Let p_1, \dots, p_k be public primes such that $\prod_{i=1}^k p_i < n$. Define the compression function $h : \{0, 1\}^k \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$ by

$$h(x, y) = y^2 \prod_{i=1}^k p_i^{x_i} \bmod n,$$

where x_i is the i th bit of x . The security of this construction relies on the so-called Very Smooth Square Root Problem, which is connected to the difficulty of factoring.

Discrete log hash. The discrete log hash, or the Chaum-van Heijst-Pfitzmann hash function [6] is defined as follows. Let p and $q = \frac{p-1}{2}$ be large, odd primes. Let α and β be randomly chosen primitive elements of \mathbb{Z}_p , such that $\log_\alpha(\beta)$ is hard to find. Define the compression function $h : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow \mathbb{Z}_p^*$ by

$$h(x, y) = \alpha^x \beta^y \bmod p.$$

Then it can be shown that a collision for h enables one to compute $\log_\alpha(\beta)$.

MASH. The MASH (for Modular Arithmetic Secure Hash) functions [15] are standardised in ISO/IEC 10118-4:1998. The compression function of MASH-1 is defined as follows. Let n be an RSA modulus, and let the message block be expanded to x , where the 4 most significant bits of every byte are set to 1111 (except in the final (padding) block, where 1010 is inserted). Let $a = \text{f00} \dots \text{00}$ (in hexadecimal), and let

$$h(x, y) = \left(((x \oplus y) \vee a)^2 \bmod n \right) \oplus y.$$

In MASH-2, the exponent 2 is replaced by $2^8 + 1$.

The MASH functions fall somewhat outside the category of provably secure hash functions, since no security proof exists. The claimed security of both these hash functions is $n^{1/2}$ for preimages, and $n^{1/4}$ for collisions.

7 Conclusion

The DAKOTA hash function is proposed. The properties of DAKOTA are that it reduces the problem of constructing a secure (collision resistant) compression function to the problem of constructing a function f such that it is infeasible to find x, \tilde{x}, z with $f(x)/f(\tilde{x}) = \pm z^2 \pmod n$, given that factoring the RSA modulus n is infeasible. The function f must be collision resistant and one-way, but it does not need to compress, and hence it can be injective.

Two proposals for the function f have been given. One combines modular arithmetic with symmetric encryption, and the other uses only symmetric encryption in the form of the AES encryption function in black-box mode.

For both versions of DAKOTA, performance is good compared to other hash functions based on modular arithmetic.

Acknowledgments

We would like to thank Dan Bernstein, Thomas Shrimpton, and the anonymous referees for helpful comments and discussions.

References

1. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
2. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and Reduced SHA-1. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)
3. Boneh, D., Franklin, M.K.: Efficient Generation of Shared RSA Keys (Extended Abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 425–439. Springer, Heidelberg (1997)
4. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
5. Charles, D., Goren, E., Lauter, K.: Cryptographic Hash Functions from Expander Graphs. In: NIST Second Cryptographic Hash Workshop, Corwin Pavilion, UCSB Santa Barbara, California, USA, August 24–25 (2006), http://csrc.nist.gov/groups/ST/hash/documents/LAUTER_HashJuly27.pdf [2008/1/14]
6. Chaum, D., van Heijst, E., Pfitzmann, B.: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (1992)
7. Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an Efficient and Provable Collision-Resistant Hash Function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)
8. Dai, W.: Crypto++® Library 5.5.2 (2007), <http://www.cryptopp.com> [2008/1/11]
9. Damgård, I.: Collision Free Hash Functions and Public Key Signature Schemes. In: Chaum, D., Price, W.L. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 203–216. Springer, Heidelberg (1988)

10. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
11. Dobbertin, H.: Cryptanalysis of MD4. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 53–69. Springer, Heidelberg (1996)
12. Finiasz, M., Gaborit, P., Sendrier, N.: Improved fast syndrome based cryptographic hash function. In: ECRYPT Hash Workshop, Barcelona, Spain, May 24–25 (2007), http://events.iaik.tugraz.at/HashWorkshop07/papers/Finiasz_ImprovedFastSyndromeBasedCryptographicHashFunction.pdf [2008/1/3]
13. The GNU MP Bignum Library (2007), <http://gmp.lib.org> [2008/3/25]
14. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing* 17(2), 281–308 (1988)
15. ISO/IEC 10118-4:1998, Information technology – Security techniques – Hash-functions – Part 4: Hash-functions using modular arithmetic
16. Kargl, A., Meyer, B., Wetzel, S.: On the Performance of Provably Secure Hashing with Elliptic Curves. *International Journal of Computer Science and Network Security* 7(10), 1–7 (2007)
17. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A Modest Proposal for FFT Hashing. In: Nyberg, K. (ed.) Fast Software Encryption 2008, Proceedings. LNCS, Springer (to appear, 2008)
18. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton (1997)
19. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
20. Montgomery, P.L.: Modular Multiplication Without Trial Division. *Mathematics of Computation* 44(170), 519–521 (1985)
21. National Institute of Standards and Technology. FIPS PUB 180-1, Secure Hash Standard, April 17 (1995)
22. National Institute of Standards and Technology. FIPS PUB 180-2, Secure Hash Standard, August 1 (2002)
23. National Institute of Standards and Technology. Special Publication 800-57. Recommendation for Key Management – Part 1: General (revised) (March 2007)
24. Rivest, R.L.: The MD5 Message-Digest Algorithm, RFC 1321 (April 1992)
25. Rivest, R.L., Shamir, A., Adleman, L.M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* 21(2), 120–126 (1978)
26. RSA Laboratories. PKCS #1: RSA Cryptography Standard (Version 2.1, June 14, 2002), <http://www.rsa.com/rsalabs/node.asp?id=2125> [2008/1/3].
27. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
28. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
29. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA?

Yevgeniy Dodis and Prashant Puniya

Department of Computer Science,
Courant Institute of Mathematical Sciences,
New York University
{dodis,puniya}@cs.nyu.edu

Abstract. Cascade chaining is a very efficient and popular mode of operation for building various kinds of cryptographic hash functions. In particular, it is the basis of the most heavily utilized SHA function family. Recently, many researchers pointed out various practical and theoretical deficiencies of this mode, which resulted in a renewed interest in building specialized modes of operations and new hash functions with better security. Unfortunately, it appears unlikely that a new hash function (say, based on a new mode of operation) would be widely adopted before being standardized, which is not expected to happen in the foreseeable future.

Instead, it seems likely that practitioners would continue to use the cascade chaining, and the SHA family in particular, and try to work around the deficiencies mentioned above. In this paper we provide a thorough treatment of how to soundly design a secure hash function H' from a given cascade-based hash function H for various cryptographic applications, such as collision-resistance, one-wayness, pseudorandomness, etc. We require each proposed construction of H' to satisfy the following “axioms”.

1. The construction consists of one or two “black-box” calls to H .
2. In particular, one is not allowed to know/use anything about the internals of H , such as modifying the initialization vector or affecting the value of the chaining variable.
3. The construction should support variable-length inputs.
4. Compared to a single evaluation of $H(M)$, the evaluation of $H'(M)$ should make at most a fixed (small constant) number of extra calls to the underlying compression function of H . In other words, the efficiency of H' is negligibly close to that of H .

We discuss several popular modes of operation satisfying the above axioms. For each such mode and for each given desired security requirement, we discuss the weakest requirement on the compression function of H which would make this mode secure. We also give the implications of these results for using existing hash functions SHA- x , where $x \in \{1, 224, 256, 384, 512\}$.

1 Introduction

The *Cascade construction* is a very elegant way to build a hash function H on arbitrary-length inputs from a given compression function h on fixed-length

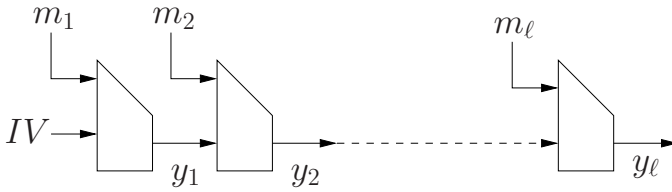


Fig. 1. The plain Merkle-Damgård Mode

inputs. Recall that for a given $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, one can define a hash function H , parametrized by an initialization vector $IV \in \{0, 1\}^n$, as follows (where input $M = m_1 \parallel \dots \parallel m_\ell$ and $m_i \in \{0, 1\}^\kappa$ for $i = 1 \dots \ell$):

$$H(m_1 \parallel \dots \parallel m_\ell) = h(m_\ell, h(\dots, h(m_1, IV) \dots))$$

We will refer to this mode, depicted in Figure 1, as the MD mode or the plain MD mode (after Merkle-Damgård).

The most abundant use of the MD mode in practice comes in the design of the industry-standard hash family SHA (which consists of several specific hash functions SHA- x , where $x \in \{1, 224, 256, 384, 512\}$). Unfortunately, despite its elegance and simplicity, the plain MD mode has several deficiencies. For instance, it does not guarantee that a “global” collision of H implies a “local” collision of the compression function h , unless one preprocesses the input into a suffix-free form before applying H [10] (the particular suffix-free encoding of appending the message length is called *MD strengthening*, and is actually used in the SHA family for this reason). More seriously, it was shown by Coron et al. [9] that even MD strengthening falls prey to the “extension attack” [4] which makes it insufficient for domain extension of random oracle. Moreover, this deficiency disqualifies the natural use of “plain MD” in the design of “pseudorandom functions” [3]. Other problems also arise when the MD mode is used in applications such as key derivation [11] and target collision-resistance (or UOWHFs [2] [5, 25]).

Apart from the issues mentioned above, several other deficiencies of the MD mode against exponential-time attacks have been discovered [15, 17]. All these deficiencies, coupled with the improved brute-force attacks on the popular SHA-1 hash function proposed recently [26, 27], suggest that it is time to design a better, more “secure” mode of operation for building a variable-length input hash function. With this purpose, NIST has been organizing several workshops dedicated to coming up with the next generation hash functions [22]. However, this process will take some time, and it does not appear that such hash functions would be standardized and widely accepted in any foreseeable future. Therefore, practitioners are “stuck” with the prospect of using existing hash functions, despite all their deficiencies. Hence, there is a pressing need to design immediate “fixes” to the MD paradigm, without changing it drastically.

¹ i.e., given $H(x)$ and any extension y , one can compute $H(x \parallel y)$ without knowing x .

² Which stands for Universal One-Way Hash Functions.

There are two aims in coming up with such “fixes” to the MD mode. The first, and so far the most popular, aim is to design a slight variant of the MD mode that provably preserves a given security property of the compression function, and to do so in the most aesthetic and efficient manner. We mention only a few of the many examples of this approach. For collision-resistance, we already mentioned the well known technique of *MD strengthening*. For another example, by viewing the initialization vector as the key and applying a *prefix-free encoding* to the message, one can obtain a variable-length input pseudorandom function from a fixed-length input pseudorandom compression function [3]. In the case of target collision-resistance, Shoup [25] designed an elegant mode for building target collision-resistant (TCR) hash functions (or UOWHFs [23]) from a TCR compression function by cleverly XORing certain masks to the internal chaining variables in the MD construction. The common feature in all these results is that one assumes *exactly the same* property from the compression function h as the desired property from the hash function H . In many cases, such as the PRF and TCR examples, this means that a “secure” mode must be sufficiently different from the plain MD so that its implementation requires a non-trivial modification to the SHA implementation. Concretely, the SHA family uses a fixed public IV (as opposed to arbitrary secret IV needed for PRFs), while in the TCR case one cannot XOR the corresponding masks without modifying the internals of SHA.

Of course, we are not saying that the required modifications are too “complicated” to be correctly implemented by a serious programmer. In fact, they are not. Our point is that, irrespective of simplicity and conceptual similarity to the existing implementations, they require one to tinker with the internals of such standard implementations. And this is not only error-prone and requiring low-level programming (which could result in less optimized implementations than those done by the experts), but goes against the whole philosophy of modular design. We do not want our security engineers to know all the low-level cryptographic details. Instead, they should understand the higher-level picture of the protocols they are trying to build, and never need to worry about existing low-level libraries.

This brings us to the second approach, where one explicitly aims to design a “secure” mode that uses only *black-box calls* to the plain MD mode.³ For instance, MD strengthening satisfies this property. Other important examples include the HMAC mode for pseudorandom functions [3] and the results for domain extension of random oracle in [9]. The attractive feature of these results is that they result in a hash function with the desired property without tinkering with the internals of SHA, and can use any off-the-shelf implementation. Moreover, all these examples also satisfy the *property-preserving* property described above, and do so without any noticeable efficiency penalties as compared to the solutions following the first approach. Concretely, at the price of one or two (or sometimes zero!) extra calls to the compression function h — which is negligible for all practical purposes —, one manages to achieve the desired goal without tinkering with the internals of the existing hash functions.

³ In practice, with MD strengthening, but we ignore this aspect for now.

OUR GOAL. Not surprisingly, we will emphasize the latter approach in coming up with “fixes” for existing hash functions. That is, we consider the question of building a hash function H' achieving a given security property P using a black-box MD-based hash function H (with an unknown compression function h). We require that the proposed construction H' satisfies the following “axioms”:

1. The construction should consist of one or two “*black-box*” calls to H . In particular, the construction is not allowed to use any knowledge of or tinker with the internals of the hash function H .
2. The construction must support variable-length inputs.
3. Compared to a single evaluation of $H(M)$, the evaluation of $H'(M)$ should make at most a fixed (small constant) number of extra calls to the underlying compression function of H . In other words, the efficiency of H' is negligibly close to that of H .

The motivation behind requiring the construction H' to satisfy these axioms is from the viewpoint of a practitioner who understands the properties of the hash function that are needed for the security of his cryptosystem, but who wants to use an off-the-shelf standardized hash function implementation without tinkering with its internals. Such a practitioner would be willing to sacrifice the *property-preserving* aspect of the “fix” in favor of a black-box implementation.

In fact, the above “axioms” leave very little freedom in choosing the modes of operation for H' . The resulting modes are essentially the *most widely-utilized* constructions appearing in practical implementations:

1. *Plain MD Construction*: This captures the notion that the application uses the hash function as it is. We will denote this mode of operation as H .
2. *Encode-then-MD Construction*: In this case, the user encodes the hash function input before applying the plain MD construction. Examples of popular encoding schemes used are suffix-free encoding and prefix-free encoding. We will refer to the corresponding constructions as the *prefix-free MD construction* H_{pre} and the *suffix-free MD construction* H_{suf} .
3. *MD-then-Chop Construction*: Here the user applies the plain MD mode and only uses part of the output while discarding the remaining bits. In particular, existing hash functions SHA-224 and SHA-384 are obtained this way from SHA-256 and SHA-512, respectively. We denote the MD-then-chop construction that chops s bits of the output as H_{chop_s} .
4. *NMAC/HMAC Construction*: The version of the NMAC construction that we consider simply composes two applications of the plain MD mode with possibly different initialization vectors IV_1 and IV_2 . While not obeying the first axiom, the NMAC construction serves as a nice abstraction for the HMAC construction which does satisfy all our axioms (but is slightly harder to formally analyze in some cases). Concretely, the HMAC construction uses the NMAC construction with $IV_1 = h(IV, \alpha_1) = H(\alpha_1)$ and $IV_2 = h(IV, \alpha_2) = H(\alpha_2)$, where each α_i is either the null string \perp (in which case we let $h(IV, \perp) = IV$) or a single κ -bit block. We denote the NMAC construction as H_{nmac} and the HMAC construction as H_{hmac} .

Now we can finally rephrase our goal as follows. Given a particular desired security property P (such as collision-resistance or pseudorandomness) and one of the 4 modes of operation above (which all satisfy our axioms), find the weakest security assumption(s) P' on the compression function h which would make the corresponding mode satisfy P (or determine that the construction is insecure for any h). Ideally, this security property P' for h would be P itself (which would result in a *property-preserving mode of operation*). However, unlike most previous work, property preservation is not our primary concern. In particular, we will not declare a mode of operation to be “insecure” for a property P simply because it is not property-preserving for P . Instead, we will find the weakest security property P' of the compression function that makes the resulting construction secure. This will allow the practitioners to decide whether or not it is reasonable to assume that the compression function of existing hash functions, such as SHA, satisfy the property P' , even if P' is (slightly) stronger than P .

OUR RESULTS. We achieve our main goal for a very wide variety of security properties including *collision-resistance (CR)*, *pseudorandomness (PR)*, *indifferentiability from random oracle (RO)*, *message authentication (MAC)*, *target collision-resistance (TCR)*, *second preimage-resistance (SPR)*, *randomness extraction (RE)* and *one-wayness (OW)*. In each case, and for each of the four popular modes above, we will identify the needed property P' on h . In some cases, the needed P' easily follows from some existing work (for instance, from [9] in the case of domain extension of random oracle). In other cases, it required some minor, but important modifications to the existing results in order to satisfy our axioms. For example, by assuming that “ $h(IV, random) = random$ ” in addition to h being a PRF when keyed with the first n bits of its input, we could build a variable length PRF using the encode-then-MD mode and adjusting the proof of [3]. More interestingly, by making extra assumptions on h , in some cases we can prove security of the modes which were previously believed “insecure” because they were not property-preserving. Finally, in some cases the proof will involve careful and non-trivial modification of previous results. For example, this is the case when analyzing the one-wayness of the H_{suf} construction.

In addition to giving an exhaustive “mode \times property” guide (see figure 2) for achieving a given security property with a given popular mode, in each section we also mention the practical implication of our results when using existing hash functions SHA- x , where $x \in \{1, 224, 256, 384, 512\}$.

RELATED WORK. We have already cited many of the relevant papers. In particular, the variants of the MD mode that are useful in the property-preservation of collision-resistance [10], pseudorandomness [3,4], message-authentication [1,21], random oracles [9] and randomness extraction [11]. We also mention the works of [7,8] concerned with multiple property-preservation; namely, designing a single mode of operation which simultaneously preserves several properties. Unfortunately, the modes of [7,8] do not satisfy our axioms. Finally, we mention the work of Halevi and Krawczyk [14], which concentrated on building TCR hash functions, and is the closest in spirit to our motivation (indeed, we will use their results when discussing the TCR property). The authors built TCR hash

	Plain MD	Encode-then-MD	MD-then-Chop	NMAC/HMAC
CRHF	(1) + (2)	Suf-Free+(1) Pre-Free+(1)+(2)	(1') + (2)	N/HMAC+(1)+(2) $\alpha_1 \neq \perp$
PRF	Append key + (1)+(2)+(4)	SF+(1)+(4) (append) PF+(2')+(3) (prepend)	Prepend key + (2')+(3')	N/H+(3)+(4) _(prepend) Any IV_s/α_s
RO	Not Secure	Suf-Free not secure Pre-Free+(5)	(5) worse security	NMAC/HMAC+(5) $IV_1 \neq IV_2 ; \alpha_1 \neq \alpha_2$
MAC	Append key + (1)+(2)+(6)	SF+(1)+(6) (append) PF+(1)+(2)+(6) _(app.)	Append key + (1)+(2)+(6')	N/H+(1)+(2)+(6) Any IV_s/α_s
TCR	$key \oplus blks$ (7) + (9)	SF+(7) ($key \oplus blks$) PF+(7)+(9)	$key \oplus blks$ (7') + (9)	N/H+(7)+(9) (append) Any IV_s/α_s ($key \oplus blks$)
SPR	(8) + (9)	SF+(9) PF+(8)+(9)	(8') + (9)	N/H+(8)+(9) Any IV_s/α_s
RExt	(10) $H_\infty(M) \wedge H_\infty(m_\ell)$	MDS + (10)(SF/PF??) $H_\infty(M) \wedge H_\infty(m_\ell)$	(10) $H_\infty(M)$	NMAC + (10) HMAC??
OWF	(2)+(11)	MDS+(2)+(11) (SF/PF??)	(2')+(11)	NMAC+(2)+(11) HMAC??

Assumptions on compression function:	Misc.
(1)=Collision Resistance (CR) (1')=CR after Chop	SF=Suffix-free
(2)=Output Regular (2')= $h(U_n, \cdot)$ is output regular	PF=Prefix-free
(3)=standard PRF (sPRF) (3')=sPRF after Chop	MDS=MD Strengthening
(4)=dual PRF (dPRF)	??=not known to be secure
(5)=FIL-RO	RExt=Randomness Extn.
(6)=MAC with κ -bit key	$Key \oplus Blks$ =XOR key to each block
(7)=enhanced SPR (eSPR) (7')=eSPR after Chop	
(8)=computed SPR (cSPR) (8')=cSPR after Chop	
(9)=Fixed-point at random IV	
(10)=Family of random functions	
(11)=One-way function	

Fig. 2. Table for comparing Security Property vs. Mode of operation

functions using the encode-then-MD mode, and showed a simple coding scheme that yields a secure TCR hash function under an appropriately strong assumption on the underlying compression function h (still weaker than CR, but stronger than TCR).

LOCATION OF THE KEY IN KEYED CONSTRUCTIONS. We note that for keyed constructions, such as constructions of pseudorandom and TCR functions, there are more than one possibilities for each hash function mode of operation. In particular, any construction for these primitives must specify the location of the key. In keeping with the black-box nature of the modes of operation, we prevent popular keying methods such as setting the key to be the IV or XORing the key into the chaining variables since this violates our basic axioms.

Moreover, we also do not consider the dedicated-key setting [18], where there is separate space for the key in each application of the compression function.

This is because existing hash functions do not support such dedicated keys. Even though we may consider the key to be part of the message block bits, we do not analyze this method since it yields constructions with poor input bandwidth (thus violating our last axiom). Hence, we will only consider modes of operation which incur an additive constant overhead compared to the plain MD mode.

ARE WE ASKING TOO MUCH? In our motivation, we advocated the fact that the security officers should not know (or worry about) the low-level details of the hash function implementations. In particular, we do not want them to manually modify the internals of SHA. On the other hand, to use our result they have to be “smart enough” to understand the purpose of their application of the hash function, so they can use our black-box workarounds. For example, they need to know if H' is used for collision-resistance, key derivation, one-wayness, etc. Aren't we asking too much? Should not the security engineer just believe that the existing hash function will be “magically applicable” for whatever intuitive use (s)he has in mind (therefore making this paper “useless”)?

We give two answers. First, we personally believe that a person designing a cryptographic protocol using a hash function *should* know what security properties this hash function should satisfy. (And this does not contradict our desire to protect them from low-level details!) Second, in order for the security engineer to use a hash function in the “magical” way above, the function should not have the weaknesses of the SHA family we mentioned earlier. Thus, until a new, “magic” hash function is built and standardized, we simply *cannot achieve* a positive answer to our question, even if we *want* our engineers to be “dumb” and not understanding what they is doing (which we personally disagree with)! Until then, we believe that the results of this paper are meaningful and useful.

2 Security of MD Modes

We will analyze each of the security properties that actual hash functions are often required to satisfy, and find the minimal assumptions on the compression function that are necessary to prove the security of each of the black-box modes of operation for this security notion. As we discussed, we will not restrict ourselves to the case of property-preservation and in some cases, we will need to make slightly stronger assumptions on the compression function than the security notion desired.

Since the focus of our paper is mostly qualitative, in terms of when (i.e. for which applications) does it make more sense to use some particular mode of operation, so we will keep the discussion “slightly informal” by using more asymptotic definitions for the security notions. We assume basic familiarity with these notions, but provide the formal definitions in the full version of this paper [12]. Due to space constraints, we only give the security of the modes of operation for collision-resistance, pseudorandomness and one-wayness in the main body. The discussion for other security notions can be found in the full version of this paper [12].

2.1 Collision Resistance

We will analyze each of the four modes for minimal assumptions required on the compression function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ needed in order to prove its collision resistance. A construction will be called ϵ collision resistant if the maximum advantage of an efficient attacker in finding a collision is ϵ . As we discussed, in some cases, the security property needed for the compression function h may be stronger than collision resistance.

PLAIN MD CONSTRUCTION. It is a well-known fact that simply assuming collision resistance of the compression function does not suffice to prove collision resistance of the plain MD construction. Indeed, if the compression function h has a *fixed-point* such that there is some $x \in \{0, 1\}^\kappa$ such that: $h(x, IV) = IV$. Then the output of the plain MD construction H collides for the inputs x and $x \parallel m$, for any m . Thus we, at least, need the compression function to satisfy the following property.

Assumption 1 (No Fixed-Points) *A function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a ϵ secure against fixed points if for a randomly chosen $IV \in \{0, 1\}^n$ no efficient machine A has success probability more than ϵ of finding a sequence of κ -bit blocks $x_1 \dots x_i$ such that,*

$$h(x_i, h(\dots, h(x_1, IV) \dots)) = IV$$

If the compression function is such that no efficient attacker can find such fixed points (along with being collision resistant), then the plain MD construction can be proven to be collision resistant. The proof of the following observation is immediate from [10].

Observation 1 *The plain MD construction can be proven to be collision resistant if the compression function is collision resistant and is secure against fixed-points.*

The *no fixed-points* assumption allows us to prove collision resistance of the plain MD construction, but it is a non-standard assumption and it is not intuitively clear as to which compression functions satisfy this property. But since we are already assuming the compression function to be collision-resistant, perhaps we can prove this result by making a weaker and cleaner additional assumption on the compression function. Fortunately we show that simply assuming output regularity suffices in this case.

Assumption 2 (Regularity of outputs) *A function $h : \{0, 1\}^m \rightarrow \{0, 1\}^n$ is a ϵ output regular function if for any efficient machine A that gives a 1 bit output:*

$$|\Pr[A(x) = 1 | x \leftarrow h(U_m)] - \Pr[A(x) = 1 | x \leftarrow U_n]| \leq \epsilon$$

Here U_m and U_n denote the uniform distributions on $\{0, 1\}^m$ and $\{0, 1\}^n$, respectively.

We show that if the compression function is output regular (i.e. for a random input, the output is well distributed over the range) in addition to being collision-resistant, then it is secure against fixed points and thus a CRHF using the observation above.

Lemma 1. *The compression function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is $(\epsilon_{col} + \epsilon_{reg} + 2^{-n})$ -secure against fixed points if it satisfies the following properties:*

- h is ϵ_{col} collision resistant.
- h is an ϵ_{reg} output regular function.

Proof: To the contrary, say there is an efficient attacker that finds a fixed point $x_1 \dots x_i$ with non-negligible probability ϵ , then we can show that it either breaks the collision resistance or the output regularity assumption for the compression function. In order to show this, choose the initialization vector IV as $IV \leftarrow h(x)$ (for $x \leftarrow U_\kappa \times U_n$), instead of $IV \leftarrow U_n$. If the success probability of A changes by a non-negligible amount then we can break the output regularity assumption. Thus, $\epsilon' \geq \epsilon_{reg} + \Pr[A \text{ succeeds in new game}]$.

To estimate the success probability of the attacker A in the new game, say it finds a sequence of κ -bit blocks $x_1 \dots x_i$ such that $h(x_i, h(\dots, h(x_1, IV) \dots)) = IV$ with probability ϵ' . Let $y = (x_i, h(\dots, h(x_1, IV) \dots))$. Then it is the case that $h(x) = h(y)$ (where x was used to select the IV). Thus, we can deduce that,

$$\begin{aligned} \epsilon' &= \Pr[(A \text{ succeeds}) \wedge (x = y)] + \Pr[(A \text{ succeeds}) \wedge (x \neq y)] \\ &\Rightarrow \epsilon' \leq \Pr[(x = y)] + \epsilon_{col} \\ &\Rightarrow \epsilon' \leq \epsilon_{col} + \sum_{IV \in \{0,1\}^n} \frac{\#\{x \text{ s.t. } h(x) = IV\}}{2^{n+\kappa}} \cdot \frac{1}{\#\{x \text{ s.t. } h(x) = IV\}} \\ &\leq \epsilon_{col} + 2^{-n} \end{aligned}$$

Thus we get that the maximum success probability of an efficient fixed-point finding attacker is $\epsilon_{reg} + \epsilon_{col} + 2^{-n}$. □

Corollary 1. *The plain MD construction H using a compression function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a $(\epsilon_{reg} + \epsilon_{col} + 2^{-n})$ collision resistant hash function if h satisfies the following properties:*

- h is ϵ_{col} collision resistant.
- h is an ϵ_{reg} output regular function.

ENCODE-THEN-MD CONSTRUCTION. It makes sense to only consider deterministic input coding schemes, since the resulting construction must behave like a function. We analyze two of the most popular such coding schemes, i.e. *prefix-free encoding* and *suffix-free encoding*.

We first note that using a prefix-free encoding on the input does not enable us to get rid of any security properties in lemma □. Hence we can essentially restate

the same result for the prefix-free MD construction H_{pre} as well. On the other hand, if we use a *suffix-free encoding* (such as Merkle-Damgård strengthening) then the resulting suffix-free MD construction H_{suf} can be shown to be collision resistant by simply assuming the collision-resistance of the compression function h [10,19].

MD-THEN-CHOP CONSTRUCTION. Note that simply assuming collision resistance of the compression function is not useful for this construction, since we truncate s bits of the output. For instance, consider the case when h is collision resistant on these s bits, and is the constant function for all other bits (noted by Kelsey [16]). However, in our setting this only means that we need to make a stronger assumption on the compression function h . In particular, we will instead assume that h is collision resistant even if we remove these s bits from its output.

Lemma 2. *The MD-then-chop construction H_{chop_s} , using a compression function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, is a $(\epsilon_{reg} + \epsilon'_{col} + 2^{n-s})$ collision resistant hash function if the following holds:*

- *The function $h' : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^{n-s}$ defined as $h'(x, y) = h(x, y)|_{n-s}$ (i.e. chopping the last s bits from the output of h) is a ϵ'_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*

The proof of this lemma is essentially the same as for corollary [□](#).

NMAC/HMAC CONSTRUCTION. We note that using the NMAC construction H_{nmac} does not help in improving upon the collision resistance of the plain MD construction H . This is essentially because any collision in the first application of the plain MD construction of H_{nmac} (using initialization vector IV_1) essentially implies a collision for the entire construction. Hence, at best, we can restate lemma [□](#) for this construction as well.

Since the HMAC construction H_{hmac} is simply a black-box instantiation of the NMAC construction, this does not help in improving collision resistance. However, we note that it has the best exact security if $\alpha_1 \neq \perp$.

2.2 Pseudorandomness

An issue in the pseudorandomness analysis of the MD modes of operation is the location of the PRF key. As discussed above, we need to specify the location of the key such that the resulting construction is still a black-box variant of plain MD. For our analysis, we will assume the key length to be the length of a single block (i.e. κ bits for the compression function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$), and we will denote the key as K . We will analyze two approaches for keying each MD mode of operation:

1. *Prepend the key to input:* The PRF construction H outputs $H(K \parallel X)$ on input X .
2. *Append the key to input:* The PRF construction H outputs $H(X \parallel K)$ on input X .

Moreover, we will need two versions of pseudorandomness definitions for the compression function, one where the key occupies the last n bits and other where it occupies the first κ bits. We get the following two assumptions on the compression function in this manner.

- *Standard PRF (sPRF) security:* Here we require that for a uniformly chosen $K \in \{0, 1\}^n$, the function $h(\cdot, K)$ must be indistinguishable from a truly random function.
- *Dual PRF (dPRF) security:* Here we require that for a uniformly chosen $K \in \{0, 1\}^\kappa$, the function $h(K, \cdot)$ must be indistinguishable from a truly random function.

Depending on the maximum distinguishing advantage ϵ of an efficient attacker in each case, we call the compression function h ϵ -sPRF or ϵ -dPRF.

PLAIN MD CONSTRUCTION. In this case if we prepend the PRF key to the hash function input, then the resulting construction is not a PRF. This is because an attacker can use the *extension attack* to find $H(K \parallel X \parallel Y)$ by simply knowing the output $H(K \parallel X)$ and computing the compression function on the remaining blocks itself (where it does not need to know the key K). On the other hand, if we append the PRF key to the input, then we can show that if the plain MD construction using h is collision-resistant and satisfies the dual PRF security, then the plain MD construction $H(\cdot \parallel K)$ is a variable-length input PRF.

Lemma 3. *The plain MD construction H is a $\mathcal{O}(\ell \cdot (\epsilon_{col} + \epsilon_{reg}) + \epsilon_{dprf})$ PRF⁴ (with PRF key appended to the function input) if the following conditions hold:*

- h is ϵ_{col} collision resistant.
- h is a ϵ_{reg} output regular function.
- h is a ϵ_{dprf} dual pseudorandom function.

The proof of this lemma is rather straightforward. Here, output regularity and collision resistance of the compression function together imply the collision resistance of the plain MD construction. Thus, in the last round, the n -bit chaining variable is different for two different inputs. Hence a distinguisher for the plain MD construction can be used directly by the dual-PRF distinguisher for the compression function.

ENCODE-THEN-MD CONSTRUCTION. Once again, we will discuss two deterministic coding schemes here, *prefix-free encoding* and *suffix-free encoding*. Let us first analyze the suffix-free MD construction H_{suf} . If we prepend the key to the (encoded) input, the resulting construction is still insecure since the *extension attack* works in this case as well. On the other hand, if we append the key to the (encoded) input then the resulting construction is a PRF if the suffix-free MD construction H_{suf} using the compression function h is a dual PRF and collision resistant (for which we only need collision resistance of h in this case).

⁴ ℓ denotes the maximum number of κ -bit blocks in a hash function input, throughout this paper

For the prefix-free MD construction H_{pre} , if we append the key to the (encoded) input then we get no advantage as compared to the plain MD construction and we can only restate lemma 3 in this case. On the other hand, if we prepend the PRF key to the (encoded) input then the resulting construction is not vulnerable to the *extension attack* in this case. Indeed, it was shown by Bellare et al. in [3] that the prefix-free MD construction with the PRF key in the IV is a PRF only assuming that the compression function h satisfies the standard PRF security. However, since we will need to prepend the key to the input (in order to preserve the black-box property of the construction), we will need to impose an extra condition on the compression function. In particular, we require that the function defined as $h(U_n, \cdot)$ is an output regular function. That is, if the first n bits of the compression function h are chosen at random then the resulting function is output regular with high probability.

Lemma 4. *The prefix-free MD construction H_{pre} is a $\mathcal{O}(\epsilon'_{reg} + \ell \cdot \epsilon_{sprf})$ secure PRF (with PRF key prepended to the input) if the following conditions hold:*

- h is a ϵ_{sprf} sPRF.
- $h(U_n, \cdot)$ is a ϵ'_{reg} output regular function.

The proof of this lemma is similar to the result of [3].

MD-THEN-CHOP CONSTRUCTION. If the PRF key is appended to the input to the MD-then-Chop construction H_{chop_s} , then a slight variant of lemma 3 can be stated for this construction as well. Indeed, all we need is to specify the dual PRF and collision-resistance properties for the compression function with chopped output.

On the other hand, if we prepend the PRF key to the input to H_{chop_s} , then the extension attack does not seem to go through as in the case of plain MD construction. This is because the attacker does not learn the chopped s bits of the chaining variable by observing the output of H_{chop_s} for the prefix of an input. Indeed, this construction can be proven to be an arbitrary-length input PRF by making a slightly non-standard assumption on the compression function. In particular, we require the compression function to satisfy the following *resilient sPRF* assumption:

Assumption 3 ((s, ϵ)-resilient sPRF) *The function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a (s, ϵ)-resilient sPRF if it is a ϵ -secure sPRF even if the attacker learns s bits of the n bit key.*

Lemma 5. *The MD-then-Chop construction H_{chop_s} is a $\mathcal{O}(\epsilon'_{reg} + \ell \cdot \epsilon'_{sprf})$ secure PRF (with PRF key prepended to the input) if the following conditions hold:*

- h is a (s, ϵ'_{sprf}) -resilient sPRF.
- $h(U_n, \cdot)$ is a ϵ'_{reg} output regular function.

NMAC/HMAC CONSTRUCTION. The NMAC and HMAC constructions were shown to be secure arbitrary-length input PRFs by Bellare [2]. In [2], it is shown

that the HMAC construction with $\alpha_1 = \alpha_2 = \perp$ (i.e. with the same IV for both invocations of the plain MD construction) is a secure arbitrary-length input PRF if the underlying compression function satisfies both the standard and dual PRF security definitions. This is done by simply prepending a different κ -bit key to each invocation of the plain MD construction [\[3\]](#).

Lemma 6. *The NMAC (resp. HMAC) construction H_{nmac} (resp. H_{hmac}) is a $\mathcal{O}(q^2\ell \cdot \epsilon_{sprf} + \epsilon_{dprf})$ PRF (with a different κ -bit key prepended to the input in each call to the MD construction) for any IV_1 and IV_2 (resp. α_1 and α_2) if the following conditions hold:*

- h is a ϵ_{sprf} -secure sPRF.
- h is a ϵ_{dprf} -secure dPRF.

2.3 One-Wayness

One way functions are also often referred to as preimage resistant functions. A construction is ϵ -secure OWF if no efficient attacker can find the input corresponding to the output of the function (on a random input) with probability more than ϵ . This security property is even weaker than second preimage resistance.

PLAIN MD CONSTRUCTION. In this case, we will need to assume that the compression function h is a one way function. Moreover, we will also require that h is output regular, so that its output is uniformly distributed for a random input. This is essentially because we need the input to a one-way function to be random in order to use the one-wayness property.

Lemma 7. *The plain MD construction H is $\mathcal{O}(\ell \cdot \epsilon_{reg} + \epsilon_{owf})$ -secure OWF if the following conditions hold:*

- h is an ϵ_{reg} output regular function.
- h is a ϵ_{owf} -secure one-way function.

The proof of this lemma is based on the fact that an attacker cannot tell the difference between the output of H on a random input or the compression function h on a random input, if h is output regular. Thus the one-wayness attacker for h can use the one for H directly.

ENCODE-THEN-MD CONSTRUCTION. If we use an arbitrary suffix-free encoding with the MD construction, then we cannot say much about one-wayness of the construction since the input distribution could be arbitrary. However, if we apply *Merkle-Damgård strengthening* to the input, then we can show that the resulting construction is a one-way function under sufficient assumptions. The proof of this fact is non-trivial though. In particular, we need to make an additional assumption about the compression function.

⁵ If the same key is prepended in both invocations, then the construction is secure under a slightly stronger assumption, called security against *related-key attacks* in [\[3,2\]](#). We ignore this setting here

Assumption 4 ((p, ϵ) **output consistent**) *The function $h : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is (p, ϵ) output consistent if for any κ -bit block x and uniformly distributed $y \in \{0, 1\}^n$, with probability at least $(1 - \epsilon)$ the number of $y' \in \{0, 1\}^n$ such that $h(x, y) = h(x, y')$ is at most p .*

Note that this property certainly holds for a random compression function (and, thus, holds for most compression functions). By making this additional assumption from the compression function, we can derive the following result.

Lemma 8. *The suffix-free MD construction H_{suf} that uses MD strengthening for suffix-freeness is $(p_{cons} \cdot (\ell \cdot \epsilon_{reg} + \epsilon_{owf}) + \epsilon_{cons})$ -secure one-way function, where ℓ is the maximum length of an inverted input provided by the OWF attacker, if the following conditions hold:*

- h is an ϵ_{reg} output regular function.
- h is a ϵ_{owf} -secure one-way function.
- h is a $(p_{cons}, \epsilon_{cons})$ output consistent.

Proof: The proof for this lemma is essentially based on the proof of lemma 7. We construct an one-wayness attacker A' for the compression function using the attacker A that has advantage ϵ in inverting H_{suf} with MD strengthening. A' gets its challenge output y and chooses a uniformly random $i \in \{1, \dots, \ell\}^n$. It then gives $z = h(\langle i \rangle, y)$ as a challenge to A .

Now A' succeeds only if the inverse z outputted by A is i -bit long. If so, then A' can proceed similar to the case on the plain MD construction in lemma 7 if the chaining variable for z in the last round, with $\langle i \rangle$ in the message block, is the challenge y . However, from our assumptions, with probability at most ϵ_{cons} there are more than p_{cons} n -bit strings y' such that $h(\langle i \rangle, y') = h(\langle i \rangle, y)$. Thus, we get that the success probability of A is at most $(p_{cons} \cdot (\ell \cdot \epsilon_{reg} + \epsilon_{owf}) + \epsilon_{cons})$. \square

As for prefix-free encoding, once again we cannot say anything general (for the same reason as above), but when prepending the message length we are essentially back to the setting of plain MD discussed above, except we need to assume that the output of the compression function on a random IV and a fixed message block is random. In particular, we note that encoding the input in any way does not help as far as one-wayness of the construction is concerned. In fact, we only need more assumptions to prove this property, as compared to the plain MD construction.

MD-THEN-CHOP CONSTRUCTION. In order to prove the one-wayness of the MD-then-Chop construction, we need to make a stronger assumption on the compression function h . In particular, we assume that h is one-way with s bits of the output chopped. Let the one-way security of the function h with truncated output be ϵ'_{owf} . Then we can show that H_{chop_s} is a $\mathcal{O}(\ell \cdot \epsilon_{reg} + \epsilon'_{owf})$ -secure one-way function (similar to lemma 7).

NMAC/HMAC CONSTRUCTION. The NMAC construction is a one-way function under the same conditions on the underlying compression function h as required in lemma 7. However, we require that random and independent initialization vectors IV_1 and IV_2 are used in the NMAC construction. However, it

turns out that translating these results to the setting of the HMAC construction is not straightforward.

3 Implications for Hash Functions in Practice

We will now translate our results into suggestions for usage of actual “cascade construction based” hash functions, such as functions from the SHA family. As we mentioned earlier, we have tried to find the minimal assumptions needed to make each of the four modes of operation secure (for each of the security properties). Thus, we have left part of the “decision making” for the practitioner who uses our results. In particular, the practitioner must consider the following questions:

1. What one needs to assume about the hash function in order for the cryptosystem (that the hash function is being used for) to be provably secure?
2. What level of trust the practitioner is willing to place in the underlying compression function?

The answer to the first question will help in deciding the security property to look for in the hash function mode of operation. The answer to the second question may not be as straightforward since the design of the compression functions is quite complex and mostly based on heuristic. In this case, the practitioner needs to weigh all the properties (s)he desires from the cryptosystem, in terms of efficiency, security etc. Thus, while some may be willing to make a slightly stronger assumption on the compression function to have a more efficient implementation, others may be willing to sacrifice some efficiency for better security. Now we will give some basic recommendations for actual hash functions with respect to the various security properties.

COLLISION RESISTANCE. Each of the SHA functions are essentially based on the suffix-free MD construction (using MD strengthening). Hence, collision resistance for each of these hash functions is asymptotically same as finding collisions on the compression function. It does not make much sense to use the “truncated” versions, SHA-224 and SHA-384, since this only sacrifices the collision resistance of the original “untruncated” version (i.e. SHA-256 and SHA-512, respectively). Using the NMAC/HMAC construction does not help in this case.

PSEUDORANDOMNESS. We note that using the full SHA-256 or SHA-512 hash functions makes more sense for pseudorandomness than using the chopped versions (SHA-228 or SHA-384), which only have worse security. If any of the SHA functions are used, as it is, for pseudorandomness, then we recommend appending the PRF key to the input instead of prepending it. However, we recommend using these functions in conjunction with a prefix-free encoding (such as prepending input length to the input) in which case the PRF key should be *preended* to the input. Another option would be to compose two calls to SHA-1, with independent keys prepended in each call, to get security based on the sPRF and dPRF security of the compression function.

RANDOM ORACLE. Note that none of the SHA functions should be used, as it is, if the security of the cryptosystem requires the *random oracle assumption* for

the hash function. This is because the plain MD construction (even with MD strengthening) is vulnerable to simple attacks in the indistinguishability scenario. One may think that both SHA-224 and SHA-384 that correspond to “chop” versions of the functions SHA-256 and SHA-512 would be secure (since the MD-then-Chop construction is secure). However, note that only 32 bits are chopped in the case of SHA-224, which does not give sufficient security for almost all applications. Hence, only SHA-384 (that chops 128 bits) may be suitable to be used directly to instantiate the random oracle.

We recommend using the HMAC construction involving two black-box calls to the SHA function (while prepending different α_1 and α_2 in each call) for this purpose. Using any of these hash functions in conjunction with a prefix-free encoding will also work for this purpose.

MESSAGE AUTHENTICATION. If the SHA functions are used as MACs directly, then the MAC key should be appended to the input. In this case, security depends on both the MAC security and collision resistance of the compression function. Using the HMAC construction does not help in improving the security either. Moreover, when the “chopped” functions SHA-224 or SHA-384 are used as MACs, then their security is only worse than the unchopped versions (SHA-256 and SHA-512).

If one is willing to assume pseudorandomness of the compression function, then the techniques mentioned above for pseudorandomness can be used as well. Another approach would be to assume the *dedicated-key setting*, by inserting the MAC key in each application of the compression function (at the cost of some input bandwidth) and then one could use one of the techniques suggested in [121].

TARGET COLLISION RESISTANCE OR UOWHFs. We recommend using the technique suggested by Halevi and Krawczyk [14] if the SHA functions are used as UOWHFs. In this case, one XORs the UOWHF key to each block of the input. Since MD strengthening is already used in all these functions, the UOWHF security of this construction is based only on the eSPR [14] (see above) of the compression function.

SECOND PREIMAGE RESISTANCE. It makes sense to use the SHA hash functions directly for the purpose of *second preimage resistance* without using any additional techniques, since they do not lead to improved security (note that these functions already incorporate MD strengthening).

RANDOMNESS EXTRACTION. All the positive results for *randomness extraction* have reasonable interpretation in practice, only if we are willing to assume that the SHA compression function is close to being a family of random functions. Even though it is theoretically impossible, since the SHA compression function has a short description, it might still be a more reasonable assumption than assuming the compression function to be a FIL-RO.

Under this assumption, we can deduce that the SHA functions are good randomness extractors for input distributions with high min entropy overall and in the last block. On the other hand, as we saw above, it might be a good idea to use chopped function SHA-384 for this purpose to get better extraction properties (SHA-224

does not have sufficient number of chopped bits to give useful advantage). Using the HMAC construction does not help in improving the extraction properties.

ONE-WAYNESS. In the case of “one-wayness”, the security of the chopped functions, SHA-224 and SHA-384, seems to rely on stronger assumptions than the security of the corresponding “unchopped” versions (SHA-256 and SHA-384). This is because the one-way security increases with the number of output bits. On the other hand, it might be the case that SHA-224 still has higher security than SHA-1, which seems intuitive given the bigger IV of SHA-224. Moreover, message encoding or HMAC construction only negatively affects the one-wayness.

4 Conclusions

In this work we showed how to efficiently use existing hash functions based on the MD mode (such as the functions in the SHA family) to build cryptographic hash functions satisfying various security properties such as collision-resistance, pseudorandomness, indistinguishability from random oracle, message authentication, target collision-resistance, second preimage-resistance, randomness extraction and one-wayness. Our constructions are black-box, support variable-length inputs and provide the same efficiency as the plain MD construction, under the minimal assumptions on the underlying compression function.

References

1. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
3. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom Functions Re-visited: The Cascade Construction and Its Concrete Security. In: Proc. 37th FOCS, pp. 514–523. IEEE, Los Alamitos (1996)
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, Springer, Heidelberg (1996)
5. Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHF's Practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, Springer, Heidelberg (1997)
6. Bellare, M., Rogaway, P.: Random oracles are practical : a paradigm for designing efficient protocols. In: Proceedings of the First Annual Conference on Computer and Communications Security, ACM, New York (1993)
7. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, Springer, Heidelberg (2006)
8. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, Springer, Heidelberg (2007)

9. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
10. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
11. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Heidelberg (2004)
12. Dodis, Y., Puniya, P.: Getting the Best Out of Existing Hash Functions or What if We Are Stuck with SHA (ful version), <http://people.csail.mit.edu/dodis/ps/sha.ps>
13. FIPS 180-1, Secure hash standard, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17 (1995) (supersedes FIPS PUB 180)
14. Halevi, S., Krawczyk, H.: Strengthening Digital Signatures Via Randomized Hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
15. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
16. Kelsey, J. In: CRYPTO 2005, Rump Session (2005)
17. Kelsey, J., Kohno, T.: Herding Hash Functions and the Nostradamus Attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
18. RFC 1321, The MD5 message-digest algorithm, Internet Request for Comments 1321, R.L. Rivest (April 1992)
19. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
20. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
21. Maurer, U.M., Sjödin, J.: Single-Key AIL-MACs from Any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
22. National Institute of Standards and Technology, NIST’s Plan for New Cryptographic Hash Functions, <http://www.csrc.nist.gov/pki/HashWorkshop/index.html>
23. Naor, M., Yung, M.: Universal One-Way Hash Functions and their Cryptographic Applications. In: STOC 1989, pp. 33–43 (1989)
24. Simon, D.R.: Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
25. Shoup, V.: A Composition Theorem for Universal One-Way Hash Functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
26. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
27. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

Replay Attack in a Fair Exchange Protocol*

Macià Mut-Puigserver, Magdalena Payeras-Capellà, Josep Lluís Ferrer-Gomila,
and Llorenç Huguet-Rotger

Universitat de les Illes Balears, Carretera de Valldemossa, km 7,5. 07120 Palma de
Mallorca, Spain

{macia.mut,mpayeras,jlferrer,l.huguet}@uib.es

Abstract. A fair multi-party exchange protocol provides equal treatment to all users, in such a way that at the end of the execution of the exchange, all parties have the element that wished to obtain, or none of them has obtained any valid item. In this paper, we analyse a well-known multi-party fair exchange protocol and, in spite of the formal proof of its correctness given in [11], we demonstrate that the protocol has a flaw. The weakness provoked by this flaw made possible a replay attack that breaks the fairness of the exchange. We will see as a group of colluding participants in the exchange can get the item from an honest participant and this participant will get nothing. In addition to that, we propose a new protocol to solve the problem of the potential replay attack which preserves the property of *semi-trusted neutral party*. The property was introduced in the original protocol so as to improve the user confidence in the trusted third party (TTP). Our solution not only preserves this property but also introduces the property of *verifiable TTP*. The property guaranties evidences from each TTP operation to the users. The evidences can be used to get compensation and correct any wrong situation caused by an incorrect operation of the TTP; for instance, in case of a passive conspiracy of the TTP.

1 Introduction

Some electronic services require an exchange of elements between two or more users. A fair exchange of values always provides an equal treatment to all users, and, at the end of the execution of the exchange, all parties have the element that wished to obtain, or the exchange has not been solved successfully (in this case, nobody has its expected element). Among the electronic applications that require a fair exchange of information we can find electronic contract signing, certified electronic mail and electronic purchase (payment in exchange for a receipt or a digital product).

In a fair exchange protocol several entities wants to exchange their goods in such a way that, at the end of the protocol, any honest participant has received

* This work was supported by ARES: Advanced Research on Information Security and Privacy (Consolider-Ingenio-2010 Program, 2007-2012), Seguridad en la contratación electrónica basada en servicios web (CICYT TSI2007 62986) and PROGECIB-16A (CAIB).

all expected items or none of them has obtained any valid item. Fair exchange protocols often use trusted third parties (TTPs) helping users to successfully realize the exchange. So, TTPs play an important role in these protocols and their reliability and security is a problem that needs to be addressed, because the security of the exchange can be broken if the TTP doesn't work properly. In this paper, we will break the fairness of the exchange of an existing protocol using a replay attack against the TTP. As a result, the protocol will finish with an entity, who has given away his own item, but without getting the other expected item.

We can distinguish between single-unit and multi-unit exchange protocols. In the case of multi-unit exchange protocols different topologies are possible (star, ring, matrix, graph). Protocols like [2, 5] suppose that each party exchanges an item against another and the exchange's topology is a ring. Franklin and Tsudik in [5] propose a new multi-party fair exchange protocol with a ring topology, which was improved in [7] and a final version of the protocol with a proof of its correctness was given in [11]. Here, we analyse the protocol proposed by Mukhamedov et al. in [11] and we will demonstrate that, in spite of the improvements suggested and the formal proof of correctness, the security of the exchange can be broken using a replay attack.

Our solution to the replay attack not only preserves a TTP with the quality of *semi-trusted neutral party* [5] but also she is *verifiable* [12]. This key feature removes a weakness of the protocol (e.g.; it can defend users against a passive conspiracy of the TTP) and increases the user's confidence in the TTP. Our new scheme will provide users with evidences of the TTP operations. In case of any misbehaviour of the TTP, participants can make use of the evidences to correct the unfair situation.

The paper is organized as follows: section 2 presents some security notions and the multi-party fair exchange protocol, section 3 describes what we can consider a reply attack, sections 4 and 5 describes the replay attack and its solution. Section 6 shows that the TTP involved in the new protocol is verifiable. Section 7 is devoted to the conclusions of this work.

2 Multi-party Fair Exchange Protocol

2.1 Syntax

To give a description of the protocols we use the notation $number.\{event\}:\{description\}$ to describe the individual steps of these protocols, where $number$ is the step number of the protocol, $\{event\}$ can be the sending of a message from user X to Y (designated by $X \rightarrow Y$) or an *ftp get* operation (designated by $X \leftrightarrow Y$) and it can also be some local computation of a participant. The notation of the encryption operations will be:

- $PU_A(m)$ is the result of applying an asymmetric encryption algorithm to the plaintext m under Alice's public key;

- $PR_A(m)$ denotes the digital signature of Alice over message m using her private key;
- $E_K(m)$ is the symmetric encryption of message m under key K .

The encryption schemes $E_K()$, $PU_A()$ and $PR_A()$ are not homomorphic.

2.2 Security Notions

In this section we will give a general definition of the security of an exchange protocol. As suggested in [10], we say that an exchange protocol is *secure* when it respect these three mandatory properties:

1. *Viability*: independently of the communication channels quality, there exists an execution of the protocol, where the exchange succeeds.
2. *Fairness*: the communication channels quality being fixed, at the end of the exchange protocol run, either all involved parties obtain their expected items or none (even a part) of the information to be exchanged with respect to the missing items is received.
3. *Timeliness*: the communication channels quality being fixed, the parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness.

So, a secure exchange protocol must be fair relating to the exchanged items and it also must be fair relating to the ability to determine the progress of the protocol (called timeliness). Moreover, a secure exchange can respect some optional properties, like non-repudiation, abuse-freeness and verifiability of the TTP [10, 12, 14]. These additional properties allow the evolution of fair-non-repudiation protocols in the critical security points, such as the trusted third party's involvement in fair exchange schemes.

2.3 Protocol Description

We will describe the multi-party fair exchange protocol proposed in [11] which will be the object of our replay attack. To do that, we need to know how the privacy and the authentication properties are achieved in this protocol, because the authors in [11], for the sake of simplicity, didn't give explicit specifications about that. For this reason we have chosen the description of the same protocol made in [7] as a way of achieving privacy and authenticity. Of course, we have added the two fixes proposed in [11] (the solution to the possible collision in the label space and the fix to the arithmetic attack that they describe).

Different topologies are possible in the case of multi-party fair exchange protocols. The exchange topology in the protocol proposed in [11] is a ring. In this case, each participant $P_i (i \in [1, n])$ desires an item (or a set of items) from the participant P_{i-1} and offers an item to the participant P_{i+1} . The communication channels between participants are unreliable and those used between each participant and the TTP are resilient (a communication channel is resilient if a message inserted into such a channel will eventually be delivered [14]).

The protocol relies on a homomorphic one-way function f (i.e.; $f(x_1 \cdot x_2) = f(x_1) \cdot f(x_2)$) and a function with n arguments F_n such that:

$$F_n(x_1, f(x_2), \dots, f(x_n)) = f(x_1 \cdot x_2 \cdot \dots \cdot x_n)$$

The proposed function f is $f(x) = x^2 \pmod{N}$ and $F_n(x_1, x_2, \dots, x_n) = x_1^2 \cdot x_2 \cdot \dots \cdot x_n$ where N is an RSA modulus. To avoid notational clutter, we suppose that all operations on subscripts are performed modulo N . As we have said, the exchange is cyclic, so, an entity $P_i, i \in [1, n]$, sends his secret information m_i to P_{i+1} in exchange of P_{i-1} 's secret information m_{i-1} . At the end of setup phase the authors suppose that:

- Each participant knows the identity of the remaining participants in the exchange;
- All participants agree on the TTP and the functions f and F_n ;
- Descriptions of the items to be exchanged $f(m_i)$ are public.

A label l has been introduced to identify a protocol's run. To avoid possible collision in the label space, Mukhamedov et al. explain that the TTP will generate and distribute the value of l to the participants of the exchange at the setup phase (timestamps or nonces may suffice for this purpose together with the TTP's name). The protocol is as follows:

1. $P_i \rightarrow P_{i+1} : PR_{P_i}(l, PU_{P_{i+1}}(R_i))$
2. $P_i \rightarrow TTP : PR_{P_i}(l, A_i, PU_T(C_i), f(R_i), f(R_{i-1}), f(m_{i-1}))$,
 where $A_i = F_n(m_i, f(m_1) \dots f(m_{i-1}), f(m_{i+1}) \dots f(m_n))$ and $C_i = m_i \cdot R_i^{-1}$
3. $TTP \rightarrow P_i : PR_T(l, \mathbf{C})$,
 where $\mathbf{C} = \{C_i \mid 1 \leq i \leq n\}$

Each participant P_i chooses a random value R_i , then she encrypts it and sends it to P_{i+1} . Then, each P_i computes $C_i = m_i \cdot R_i^{-1}$ and A_i and she sends this information to the TTP along with $f(R_i), f(R_{i-1})$ and $f(m_{i-1})$. The TTP waits until it has received the n messages from the participants and then she performs the following checks:

- a. The $f(R_i)$ sent by P_i is equal to the $f(R_i)$ sent by P_{i+1} ;
- b. All received A_i are equal. If so, the TTP computes $\zeta = C_1 \cdot \dots \cdot C_n$ and $F_{n+1}(\zeta, f(R_1), \dots, f(R_n))$. This second computation has to be equal to any A_i . These operations are made by the TTP to ensure consistency between m_i and $f(m_i)$;
- c. The TTP must be sure that C_i contains R_i^{-1} . So, the TTP checks that each $C_i \in \mathbf{C}$ verifies: $f(C_i \cdot R_i) = f(m_i), \forall i \in [1, n]$.

If checks succeed then the TTP will send \mathbf{C} to the participants. This enables each P_i to compute $m_{i-1} = C_{i-1} \cdot R_{i-1}$.

3 Replay Attacks

Replay attacks have been discussed for quite some time in the literature (e.g.; [1, 6, 13]). A replay attack is a kind of active attack where a user (the attacker) records a communication session or part of it and then, later, replays the entire session, or a portion of the recorded information so as to take advantage of it.

The attacker can be a single user involved in the exchange (e.g.; a sender or a receiver of a certified electronic mail, a payer or a payee in an electronic purchase, a contract signer...) or a confabulation of users.

Attackers want to have a benefit replaying the exchange (or part of it). This benefit, in case of fair exchange, can be obtaining the token they wanted to receive through the exchange. In order to get the element, the attacker usually contacts with the TTP and sends him some information included in the recorded exchange. As a result of this attack, the attackers gain the element they desired to obtain and some other honest participant will not receive the expected element.

4 Replay Attack to the Multi-party Fair Exchange Protocol

A replay attack against the TTP could be possible in the protocol defined in §2.3. In this attack, a group of dishonest participants make the TTP believe that the exchange is stopped. Then, they start a new protocol's run without an honest participant of the previous exchange and they use the TTP to get the item of this participant from the first exchange. We'll attempt to illustrate the attack by giving this example:

1. Suppose we have three participants involved in a cyclic exchange using the protocol specified in [11]: P_i sends her information m_i to P_{i+1} in exchange of P_{i-1} 's secret information m_{i-1} ($\forall i = 1..3$); to close the ring P_3 sends her information to P_1 . We assume that P_1 collude with P_2 against P_3 .
2. The second step of the protocol will not be performed neither by P_1 nor by P_2 . However, they capture the message sent by P_3 to the TTP:

$$PR_{P_3}(l, A_3, PU_T(C_3), f(R_3), f(R_2), f(m_2))$$

The TTP will not receive any other message of this protocol's run, so she eventually stops this protocol's instance.

3. Afterwards P_1, P_2 and P_4 (a new dishonest participant colluded with P_1 and P_2) start a new protocol's run. The target of this new exchange is getting the item m_3 of P_3 from the previous exchange. In this case, P_1 and P_2 are supposed to exchange new items (the description of the items are $f(\tilde{m}_1)$ and $f(\tilde{m}_2)$ respectively), but these participants deceive the TTP, they say that the description of the item exchanged by P_4 is $f(m_3)$. Thus, after the exchange of the random numbers at the first step of the protocol, the second step will be as follows:

$$\begin{aligned}
 P_1 &\rightarrow TTP : PR_{P_1}(\tilde{l}, \tilde{A}_1, PU_T(\tilde{C}_1), f(\tilde{R}_1), f(R_3), f(m_3)) \\
 P_2 &\rightarrow TTP : PR_{P_2}(\tilde{l}, \tilde{A}_2, PU_T(\tilde{C}_2), f(\tilde{R}_2), f(\tilde{R}_1), f(\tilde{m}_1)) \\
 P_4 &\rightarrow TTP : PR_{P_4}(\tilde{l}, \tilde{A}_1, PU_T(C_3), f(R_3), f(\tilde{R}_2), f(\tilde{m}_2))
 \end{aligned}$$

The information $PU_T(C_3), f(R_3)$ and $f(m_3)$, used in the new messages, is taken from the message captured in the previous protocol's run. The value of $\tilde{A}_{i=1,2,4}$ must be the same, so P_4 copies this value from P_1 because it is impossible to be computed by P_4 without knowing m_3 .

4. Finally, the TTP checks:
 - (a) The $f(R_i)$ sent by P_i is equal to the $f(R_i)$ sent by P_{i+1} ;
 - (b) All received A_i are equal. The TTP computes $\zeta = \tilde{C}_1 \cdot \tilde{C}_2 \cdot C_3$ and checks that $F_4(\zeta, f(\tilde{R}_1), f(\tilde{R}_2), f(R_3))$ is equal to any A_i ;
 - (c) The TTP also has to check that each $C_i \in \mathbf{C}$ verifies: $f(C_i \cdot R_i) = f(m_i), \forall i \in [1, n]$; where $\mathbf{C} = \{C_i \mid 1 \leq i \leq n\}$.
5. Participants will receive \mathbf{C} from the TTP. Participant P_1 has received R_3 in the first step of the previous protocol's run. Thus, P_1 can compute the message $m_3 = C_3 \cdot R_3$.

Following these five steps, participant P_1 has got m_3 from P_3 and P_3 has not received the message expected. So, the fairness of the exchange has been broken.

5 Solving the Replay Attack

5.1 Fixing the Protocol

We will modify the original protocol so as to solve the problem of the replay attack. As a result, we will get a new protocol resistant to this kind of attacks. In the new scheme, before starting the exchange phase of the protocol, there is a setup phase. At the end of this phase:

- a. Participants in the exchange have agreed on the identity of the TTP and on the functions f and F_n .
- b. Each participant knows the identity of the remaining participants in the exchange. The TTP knows these participants and publishes a list with their identities ($\rho = PR_T(l, P_1, P_2, \dots, P_n)$ is the set of all participants with the label of the exchange l , which identifies a protocol's run). The order of appearance in this list determines the way in which participants are arranged in the ring topology of the exchange.
- c. Participants also send the description of items that will be exchanged to the TTP ($f(m_i), \forall i \in [1, n]$). The TTP publishes a list with these descriptions: $\delta = PR_T(l, f(m_1), f(m_2), \dots, f(m_n))$.

Thus, at the end of the setup phase, participants can download from a public directory managed by the TTP the following items:

0. $P_i \leftrightarrow TTP : \rho, \delta$

Then, each user P_i verifies the correctness of (ρ, δ) and begins the exchange phase by choosing a random value R_i and sending it to P_{i+1} as follows:

1. $P_i \rightarrow P_{i+1} : PR_{P_i}(PU_{P_{i+1}}(l, R_i))$
2. $P_i \rightarrow TTP : PR_{P_i}(l, A_i, E_{K_i}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1})),$
where:

$$C_i = m_i \cdot R_i^{-1}, \text{ and}$$

$$A_i = F_n(m_i, f(m_1), \dots, f(m_{i-1}), f(m_{i+1}), \dots, f(m_n))$$

Note that label l is encrypted along with the random number R_i , the key K_i and the message C_i . Thus, nobody can use R_i or C_i outside the scope of this exchange. Inside the messages sent by participants to the TTP, $f(m_{i-1})$ was introduced in [11] because the TTP needs to know agent-message correspondence. But now, this information is known by the TTP during the setup phase (in fact, TTP publishes ρ and δ).

In order to know the status of the exchange and to verify the task of the TTP, the TTP publishes, every period of time Δt (this period is a public parameter of the TTP), an authenticated list (λ) of received messages from the participants by the TTP. The items issued by the TTP are published in a public directory managed by this entity. When all messages are received, the TTP verifies that these messages have the correct format with the appropriate label, items, signatures and encryptions. Obviously, that is very important to detect any kind of attack, but, here, we want to emphasize the role of the correctness of these verifications in messages $PR_{P_i}(l, A_i, E_{K_i}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1}))$ in order to prevent the replay attack: any participant cannot separate C_i from l , which was the basis of the replay attack of §4.

Next, the TTP performs the verifications specified in §2.3 at the step three of the protocol and, if all checks succeed, she publishes \mathbf{C} and \mathbf{K} as follows:

3. $P_i \leftrightarrow TTP : PR_T(l, \mathbf{C} = \{C_i | 1 \leq i \leq n\}, \mathbf{K} = \{K_i | 1 \leq i \leq n\})$

Finally, each participant P_i computes $m_{i-1} = C_{i-1} \cdot R_{i-1}$ and obtains the expected item. However, an execution of the protocol can be aborted by any participant if the TTP has not received the corresponding messages (step 2) from all participants and, therefore, she has not published \mathbf{C} (step 3). To cancel an exchange, any participant has to verify that the TTP has not collected all messages (i.e. the last published list λ doesn't contain all messages and \mathbf{C} is not published), then she can perform this alternative step three:

3. $P_i \rightarrow TTP : PR_{P_i}(l, \text{"cancel"})$
4. $P_i \leftrightarrow TTP : PR_T(l, \lambda, \text{"cancelled"})$

When the TTP receives a message to cancel an exchange from a participant, she checks whether the last published λ has all messages from the participants. If the check fails, the TTP makes public the message $PR_T(l, \lambda, \text{"cancelled"})$ instead of publishing the next list of received messages λ . Obviously, the list

λ in message $PR_T(l, \lambda, \text{"cancelled"})$ must not contain all messages of the step 2 but it has to contain the request to cancel the exchange: $PR_{P_i}(l, \text{"cancel"})$. Lastly, participants can get the message $PR_T(l, \lambda, \text{"cancelled"})$, which is an evidence that prove that the exchange identified by label l has been aborted.

5.2 Security of the Protocol

With respect to the three mandatory properties of a secure fair exchange we can state that this new scheme is secure because:

1. Obviously the three step of the new protocol has an execution where the exchange succeeds. So, the protocol is *viable*.
2. According to the definitions in §2.2, the proposed protocol is fair. A protocol's run always ends with a fair situation:
 - (a) If everyone is honest, then each P_i can compute: $m_{i-1} = C_{i-1} \cdot R_{i-1}$, at the end of the protocol.
 - (b) If all parties are honest but the TTP is not, all R_i values are pre-distributed securely and the TTP cannot obtain any R_i . Moreover, if the TTP misbehaves, participants will have evidences to prove that and to correct the unfair situation (see §6: verifiable third party).
 - (c) If a protocol run is aborted, a malicious participant cannot take advantage of that. Any participant will not be able to use the information of this protocol's run outside the scope of the exchange (the information items used in the protocol are linked to the exchange and the encryptions algorithms are not homomorphic, so it is not possible to use any piece of information from a particular exchange in another exchange without being detected). Thus, in this case, the malicious participant cannot learn the secret information from another participant.
3. Participants has the ability to determine the progress of the protocol and they can arrive, in finite amount of time, at a point where the protocol stops preserving fairness (here, we suppose that the TTP will reply to any message from a participant in a finite amount of time). For example, if the TTP doesn't publish **C** and **K** because she has not received all messages from participants, then any participant can cancel the protocol preserving the security of the exchange.

6 Verifiable Third Party

Besides the compulsory properties, some other remarkable properties can be present in a fair exchange. In order to improve the user confidence in the TTP, the original protocol was presented in [5] with an important property: the TTP is a *semi-trusted neutral party*. Numerous papers [3, 4, 5, 8, 9] express the concern about that, because it is necessary the user confidence in the critical security points in order to spread the use of new electronic procedures. From this point of view, we need to design new protocols where the user confidence on what

is happening when he invokes the TTP is enhanced. Otherwise, it could be very difficult to find designated full-time neutral parties to rely on in order to use any protocol.

A *semi-trusted neutral party*, as it is explained in [5], can be selected on a case-by-case basis and asked to aid in the execution of a fair multi-party exchange. However, while a TTP with this property is trusted to ensure the fairness of the exchange, it is not trusted with the actual items involved in the exchange. This means that a malicious semi-trusted neutral party must be unable to cheat as long as the other parties remain honest (i.e. a malicious TTP cannot know and reveal the content of the exchanged items).

Our solution, the proposed protocol in §5.1, is designed to achieve another property in order to increase the user confidence on the TTP operations. The TTP involved in this protocol is *verifiable* according to the concepts defined in [12]. A *verifiable TTP* aims to reduce the amount of trust that users have to place in TTPs, because *verifiability* is a security capability, that is to say, it is a mechanism to protect users against security threats. Now, we are going to give a definition of a verifiable TTP according to the concepts given in [12]:

Definition 1. *A security service is verifiable if the user, who sends a request to a TTP, receives a non-repudiation evidence of each operation carried out by the TTP to provide the service.*

Definition 2. *The verifiability of a security service is on-line if the user, after checking the received evidences, can immediately know whether the TTP misbehaved. In case of problems, the user can start a dispute to correct the situation.*

Definition 3. *A TTP is verifiable if the security services it provides are verifiable and the verifiability of the services is on-line.*

Thus, to be verifiable, a TTP has to issue evidences about its operation. For instance, if the fairness of the exchange is lost due to an incorrect TTP operation, then users will have evidences to start a dispute in order to restore the fairness.

To specify the protocol of the above section, we have followed the guidelines given in [12] so as to have a verifiable TTP operation. As a result, the TTP not only is a *semi-trusted neutral party* (she doesn't know any R_i to decrypt messages) but also is *verifiable* (any wrong operation of the TTP can immediately be detected and participants have evidences of this wrong operation. The proof can be used to achieve fairness in an external dispute resolution system).

6.1 Proof of the Verifiability

In order to show that the TTP is verifiable, we have to prove that the service it provides it's also verifiable. Previous to that, at the step 0 of the protocol the TTP issues lists ρ and δ . The lists are authentic (they have the TTP's signature). Anyone can check the correctness of the participants in the ring topology of the exchange and also the correctness of the exchanged item descriptions. Any participant, who does not agree with these lists, must not start with the protocol. She has to inform the rest of participants and the TTP about that.

Proposition 1. *The TTP involved in the protocol specified in §5.1 is verifiable.*

Claim 1. The activity performed by the TTP at the second step of the protocol is on-line verifiable.

Proof. The TTP receives the messages from each participant. Every period of time Δt , the TTP publishes the received messages during the interval in the list λ . Anyone can get λ and, in case of communication problems, any participant can know if the TTP has received his message or not. We have to remember that the channel is resilient and there is no deadline in the protocol, thus any message sent to the TTP will eventually be delivered to this entity. λ is a non-repudiation of reception evidence of the user's messages issued by the TTP. Thus, anyone can claim that a received valid message (a message made in accordance with the protocol specifications) and its information are not in the final message published by the TTP where there is \mathbf{C} . Each user has the following evidences to solve a dispute about this activity: ρ, δ, λ , its message $(PR_{P_i}(l, A_i, E_{K_i}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1})))$ and the final message of the TTP $(PR_T(l, \mathbf{C}, \mathbf{K}))$.

Claim 2. At the step three the TTP publishes the list of encrypted messages \mathbf{C} . According to the protocol specified in §5.1 this is an on-line verifiable activity.

Proof. As we have seen in claim 1, any user has enough evidences to prove that all encrypted messages from participants have to be in this list and in the same order that is was specified at the setup phase (as stated in the ring topology of the participants). Moreover, the corresponding decrypted messages have to agree with the description established at the setup phase. Otherwise the TTP shouldn't have published it in the list, however users have the following evidences to claim for that in a dispute resolution system: $\rho, \delta, \lambda, PR_T(l, \mathbf{C}, \mathbf{K})$.

Claim 3. The operation of the TTP when publishes a cancellation token of an execution of the protocol is on-line verifiable.

Proof. participants of the exchange are able to check if the TTP has received all messages of the step 2 or not (if the TTP publishes any list λ with all these messages then she cannot cancel the protocol because users will have evidences of the incorrect operation). The message $PR_T(l, \lambda, \text{"cancelled"})$ is an evidence that proves that the TTP has cancelled the exchange because she has received a request to cancel the execution of the protocol from a participant.

Result. According to claims 1, 2 and 3, the security activities performed by the TTP to provide the service are on-line verifiable. Thus, the TTP involved in the protocol is verifiable: users will get non-repudiation evidences of the TTP operations. These evidences allows users to verify the service provided by the TTP, that it means that they can immediately verify the items published by the TTP to provide the service and, in case of any incorrect action that breaks the fairness of the exchange, users will have evidences to correct the situation.

6.2 Defending Against Passive Conspiracies in Case of Verifiable TTP

The protocol designed by Franklin and Tsudik in [5] has a *semi-trusted neutral party* which guarantees the message confidentiality to the users, in the sense that the TTP will not be able to know the content of items exchanged. Thus, we can say that the TTP only has an *operational* role in the protocol. In addition to that, we not only have improved the security of the protocol but also we have introduced the verifiability of the TTP as an additional property of the protocol. A *passive conspiracy* (PC) occurs whenever a dishonest party (or a group thereof) conspires with an honest party without the latter's consent. One of the possible PC scenarios of two-party fair exchange, which the TTP is involved as the attacker, is:

- The TTP could (without any consent) favor P_1 over P_2 and cause to learn m_2 without P_2 learning m_1 .

Franklin and Tsudik in [5] consider two sub-types of this kind of PC for the multi-party fair exchange:

1. The TTP selectively broadcast (i.e., directs its message to some participants but not to others).
2. The TTP intentionally sends corrupt \mathbf{C} (i.e., some C_i values are genuine and some are fake).

In order to prevent an honest participant from becoming an unwilling co-conspirator of the TTP, the authors in [5] propose a *modus operandi* for an honest participant in an exchange:

- Each participant P_i , before searching for its barter secret in \mathbf{C} , computes the product of all elements in \mathbf{C} :

$$\zeta = H(\mathbf{C}) = C_1 \cdots \cdots C_n$$

- Next, P_i checks:

$$A_i = F_{n+1}(\zeta, f(m_1), \dots, f(m_n))$$

- If the check fails, then
 - ▷ An honest P_i must halt the protocol and not compute m_{i-1} .
- Otherwise,
 - ▷ P_i computes m_{i-1} and broadcasts \mathbf{C} to all other participants (for the benefit who may not have received it, perhaps because of a misbehaving TTP).

As is said in [5], this procedure ensures that participant P_i does not learn its barter secret m_{i-1} while some other participant is denied the opportunity to learn its secret.

However, the TTP involved in the protocol specified in §2.3 is not verifiable. So, participants cannot be sure about the cause of the PC attack described at the previous paragraph, because the check $A_i = F_{n+1}(\zeta, f(m_1), \dots, f(m_n))$ can be correct even if C_i 's are untidy in \mathbf{C} . Thus, the solution proposed in [5] it is not correct because only participants, who cannot decrypt her expected message from \mathbf{C} , know that the TTP has not provided the service as the protocol specifies but they don't have any evidence to prove it (they only have: $PR_{P_i}(l, PU_{P_{i+1}}(R_i))$, and $PR_T(l, \mathbf{C})$).

Even these participants cannot say who is responsible for the attack. Because, they don't have evidences to prove whether the problem arise when some colluded participants exchanged their C_i 's before sending them to the TTP and the TTP doesn't perform all checks previous publishing \mathbf{C} or all participants are honest but the TTP misbehaves when publishes a list \mathbf{C} where some C_i 's are fake.

The problem of a dishonest TTP in the proposed PC scenario can have a different solution if the TTP is verifiable. Using the protocol of §5.1, each participant will have the following non-repudiation evidences:

- ▷ $PR_{P_{i-1}}(PU_{P_i}(l, R_i))$,
- ▷ $PR_{P_i}(l, A_i, E_{K_i}(l, C_i), PU_T(l, K_i), f(K_i), f(R_i), f(R_{i-1}))$,
- ▷ λ, ρ, δ and
- ▷ $PR_T(l, \mathbf{C}, \mathbf{K})$

This enables participants not only to verify the correctness of the TTP operations but also they can demonstrate that the TTP has misbehaved if a conspiracy of the above proposed scenario occurs:

- P_i can check that the result of applying f to K_{i+1} published in \mathbf{K} is equal to $f(K_{i+1})$ in the P_{i+1} 's message that appears in λ .
- P_i can also verify that the decryption of $E_{K_{i+1}}(l, C_{i+1})$ with K_{i+1} has the label of the exchange and the same C_{i+1} that is in \mathbf{C} .

Thus, the operation of the TTP has been verified by participants and if the TTP publishes any C_i or K_i different from the originals sent by P_i , then participants will detect this misbehaviour and they have evidences to prove it to an external arbiter (e.g. a judge).

7 Conclusions

Franklin and Tsudik defined a multi-party exchange protocol in [5]. The protocol was improved in [7] and recently was published with a formal proof of correctness using the strand space formalism [11]. In this paper, we analyse the protocol and, in spite of the improvements and the formal proof of its correctness, we have demonstrated that the protocol had a flaw. The weakness provoked by this flaw made possible a replay attack that breaks the fairness of the exchange. In §4, we

have seen as a group of colluding participants in a multi-party fair exchange can get the item from an honest participant and this participant will get nothing. To break the fairness of the exchange, dishonest participants use what we call a replay attack. A replay attack is possible in this protocol because some items of the messages exchanged in an instance of the protocol can be used in a different instance of the protocol. Then, the dishonest participants use the TTP involved in the protocol as an oracle to achieve the secret information from the honest one. The attack is possible because the private channel between participants and TTP protects the information that must be secret but this information is not linked to a particular run of the protocol.

In §5.1 we have repaired the protocol to solve the problem of the replay attack. In addition to that, the fixed protocol has a procedure which allows users to cancel an exchange. With this procedure we have added the property of *timeliness* to the protocol, then in §5.2 we have been able to demonstrate the security of the new exchange scheme. We think that the proposed fair exchange scheme is a good example how a protocol can be executed over an insecure network and provides an additional property of TTP-verifiability, i.e. a misbehaving TTP can be proven (of course the protocol also has the so-called mandatory properties of a secure fair exchange: viability, fairness and timeliness).

The role of third parties in protocols is very important. In fact, the original protocol in [5] was proposed with an important property related to the TTP: the entity is called *semi-trusted neutral party* because she cannot reveal the content of the exchanged items. The property wants to decrease a potential risk of the protocol in case of an incorrect TTP behaviour (malicious or not) and, as a consequence, the protocol expects to reduce the amount of users reluctant to use the new electronic procedures. Following this idea, our solution to replay attack has been designed to involve a verifiable third party.

The introduction of the verifiability property in security protocols aims to spread the use of such protocols. We think that this property builds a trusted infrastructure so as to enhance the user confidence on what is happening during every protocol's run. Thus, the verifiability of the third party is a security property, which can be employed to convince reluctant users in using the new electronic procedures. With verifiable TTPs we can show how a potential weakness of some security protocols can be removed, thereby improving the security of the system. Verifiable third party is a property that provides evidences to the users from each TTP operation in an easy and comprehensible way. The evidences can be used to get compensation and correct a wrong situation caused by an incorrect operation of the TTP in an external dispute resolution system. For instance, in this paper, we have seen as this property protects users against a passive conspiracy of the TTP which pretends to break the fairness of the exchange.

In further works we have to consider the way of formally proving the correctness of a protocol, because, here, we have broken the security of a fair exchange protocol with a formal proof of correctness presented in [11] (the given security proof is only related to the fairness property not with other security properties).

References

1. Aura, T.: Strategies against replay attacks. In: 10th IEEE Computer Society Foundations Workshop (CSFW 1997), pp. 59–68. IEEE Computer Society Press, Los Alamitos (1997)
2. Bao, F., Deng, R.H., Nguyen, K.Q., Varadharanjan, V.: Multi-party Fair Exchange with an Off-line Trusted Neutral Party. In: DEXA 1999 Workshop on Electronic Commerce and Security, Italy (1999)
3. European Commission Information Society DG. Open Information Interchange (OII) service: OII Guide to Trust Services, <http://www.diffuse.org/oii/en/trust.html>
4. European Telecommunications Standard Institute (ETSI): Telecommunications Security; Trusted Third Parties (TTP); Requirements for TTP Services; ETSI Guide EG 2001 057 v1.1.2 (1997-2007)
5. Franklin, M.K., Tsudik, G.: Secure Group Barter: Multi-Party Fair Exchange with semitrusted neutral parties. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 90–102. Springer, Heidelberg (1998)
6. Gong, L., Syverson, P.: Fail-stop protocols: An approach to designing secure protocols. In: 5th International Working Conference on Dependable Computing for Critical Applications, pp. 44–55 (1995)
7. Gonzalez-Deleito, N., Markowitch, O.: Exclusion-freeness in Multi-party Exchange Protocols. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 200–209. Springer, Heidelberg (2002)
8. Internet Policy Institute. Report of the National Workshop on Internet Voting: Issues and Research Agenda (March 2001), <http://www.internetpolicy.org>
9. ITU-T: Recommendation X.842: Information technology – Security techniques – Guidelines on the use and management of trusted third party services (October 2000)
10. Markowitch, O., Gollmann, D., Kremer, S.: On fairness in exchange protocols. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 451–464. Springer, Heidelberg (2003)
11. Mukhamedov, A., Kremer, S., Ritter, E.: Analysis of a Multi-Party Fair Exchange Protocol and Formal Proof of Correctness in the Strand Space model. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 225–269. Springer, Heidelberg (2005)
12. Mut Puigserver, M., Ferrer Gomila, J.L., Huguet i Rotger, L.: Certified e-mail Protocol with Verifiable Third Party. In: IEEE International Conference on e-Technology, e-Commerce and e-Service EEE 2005, Hong Kong, pp. 548–551 (2005)
13. Syverson, P.: A Taxonomy of replay attacks. In: 10th IEEE Computer Society Foundations Workshop (CSFW 1997), pp. 187–191. IEEE Computer Society Press, Los Alamitos (1997)
14. Zhou, J., Deng, R.H., Bao, F.: Evolution of Fair Non-repudiation with TTP. In: Pieprzyk, J.P., Safavi-Naini, R., Seberry, J. (eds.) ACISP 1999. LNCS, vol. 1587, pp. 258–269. Springer, Heidelberg (1999)

Improved Conditional E-Payments

Marina Blanton

Department of Computer Science and Engineering
University of Notre Dame
mblanton@cse.nd.edu

Abstract. Conditional e-cash or conditional e-payments have been introduced by Shi et al. as the means for enabling electronic payments to be based on the outcome of a certain condition not known in advance. In this framework, a payer obtains an electronic coin and can transfer it to a payee under a certain condition. Once the outcome of the condition is known, if it was favorable to the payee, the payee can deposit the coin; otherwise, the payer keeps the money. In this work, we formalize conditional payments and give a scheme to achieve conditional e-payments that outperforms the original solution in several respects.

1 Introduction

Recently Shi et al. [24] introduced conditional electronic payments which allow a participant to anonymously cash bank-issued electronic coin at a future time if a certain agreed-upon condition is satisfied. That is, a payer engages in a protocol with a payee after which the payee has a (conditional) payment that can later be cashed only if a certain public condition is satisfied (or a certain event happens). Such scenarios arise in several contexts including, for instance, trading of financial securities and prediction markets. The conditional nature of electronic payments can, however, be taken more broadly with the outcome of the condition determined by the performance of the payee in carrying out a certain task or by a combination of different conditions.

Such conditional payments have similarities with traditional e-cash systems, but the requirements placed on interaction between entities in a conditional e-payment protocol are different enough for an e-cash scheme to be adopted to this problem. Thus, new tools need to be developed to meet the requirements of conditional payments. To illustrate why e-cash solutions are not sufficient for conditional e-payments, we list some distinctive features of conditional e-cash next. In conditional payments, a payer obtains an electronic coin and anonymously transfers it to an anonymous payee. In traditional e-cash systems, however, the coin is normally bound to the identity of the merchant during the transfer, and the merchant cannot remain anonymous. Furthermore, in conditional payments the payee should be unable to spend the coin until after the outcome of the condition is determined and only if the outcome is favorable to the payee. Additionally, the payer should have the ability to cash the payment in case of an unfavorable to the payee outcome of the condition, which cannot be done in

traditional e-cash. Thus, existing e-cash schemes do not provide an adequate solution to the conditional e-payment setting with the above requirements.

Shi et al. [24] defined the model and necessary properties for conditional e-payments, as well gave the first solution to the problem. Unfortunately, this solution lacks efficiency due to the use of expensive cut-and-choose techniques, and in this work we show that recent advances in electronic cash systems allow conditional payments to be implemented in a more efficient manner completely avoiding cut-and-choose techniques. More precisely, the solution of [24] requires $O(n_1 n_2 k)$ computation and communication, where n_1 and n_2 are cut-and-choose parameters and k is a security parameter for RSA-based systems. Recall that in cut-and-choose techniques with a parameter n a dishonest user can cheat with probability $1/n$, therefore a protocol that has the overhead of $O(n^2 k)$ is likely to be too computation and communication heavy for practical use, which we remedy in this work with a faster solution. In our solution, probability of cheating drops exponentially with the increase in computation and communication. More precisely, we achieve $O(k' \log n_2)$, where k' is a security parameter for groups with bilinear maps, which significantly lowers $O(n_1 n_2 k)$. (The logarithmic factor in our solution is due to the use of verifiable encryption.)

Our contributions. The basis of construction is an e-cash system of Camenisch and Lysyanskaya that follows from CL-signatures with protocols. We modify it to (i) permit payers and payees to stay anonymous during the transfer protocol while maintaining the ability of the bank to trace dishonest payers, and to (ii) incorporate the conditional nature of the transfer. Compared to the solution of Shi et al., our simple scheme has the following advantages:

- Our scheme has lower computation and communication overhead.
- Shi et al.’s solution requires the payee to contact the bank at the time of coin transfer to verify the validity of the coin, while in our scheme all transfers between the payer and the payee are performed off-line, with no participation of the bank or other entities.
- Shi et al.’s solution gave only informal arguments regarding the security of the solution. In this work, we formally state security requirements for a conditional e-payment scheme and provide proofs of security.

Another important contribution of this work is an extension that permits payees to further transfer (conditional) payments to other payees. In this case, double-spending by both payers and payees is addressed.

The rest of this paper is organized as follows. We first give a more detailed description of the model in section 2 and list preliminaries and building blocks in section 3. Our conditional payment scheme is presented in section 4. An extension for handling additional transfers is given in section 5. And section 6 concludes the paper.

2 The Model

A conditional e-payment scheme involves several parties, namely: a bank that issues electronic coins; a publisher that announces public conditions and later

their outcomes; a payer who obtains an electronic coin from a bank and can conditionally transfer it to a payee. In this model, the payer withdraws a certain amount of money from his bank account and obtains an electronic coin for the amount of the withdraw. The publisher announces future events by posting information about them. The payer can conditionally transfer his coin to a payee using the public conditions announced by the publisher. When the outcome of the public condition becomes known, the publisher is trusted to correctly publish the outcome of the event and any other information associated with it. After the event, the payee will be able to validate her coin and cash it only if the published outcome of the event was favorable to her. Otherwise, the payee's coin remains unspendable, and the payer cashes it back.

In the rest of the paper, we use the following terminology: when the payer contacts the bank to have an electronic payment issued to him, we will refer to the token that the payer receives from the bank as a coin. Once a coin is transferred to a payee, we will refer to the token that the payee receives as a conditional payment. Once the outcome of the condition on the payee's conditional payment is announced, it can be transferred into a validated (or casheable) coin. Finally, in the event of unfavorable outcome of the condition, the conditional payment becomes uncasheable.

The properties that the conditional e-cash system in [24] was (informally) shown to have are as follows:

Anonymity. The bank is unable to associate its previously issued coins with the identities of principals cashing them (payers or payees).

Double spending. It is either infeasible to achieve or the identity of a payer will be uncovered if (i) the payer transfers a coin to more than one payee and the payees cash it or (ii) one payee cashes the coin and the payer cashes it as well.

Conditional transfer. In the case of an unfavorable outcome, the payer can cash back his coin. If the payee accepts the coin during the transfer protocol, in the case of a favorable outcome, she will be able to cash the coin.

Deniability. Neither payer nor the payee can prove to outside parties that they participated in a conditional payment protocol.

Limited information flow. The bank cannot infer any event-specific details. The publisher cannot infer any information about bank-payer-payee interactions through the protocol.

In some interactions in the conditional payments scheme, a participant is to stay anonymous. We then assume that in such cases the participant will use a network anonymizer (e.g., [16]) to hide her location information or engage in a protocol using other anonymous means (e.g., a protocol between the bank and an anonymous user can take place at a bank's kiosk).

We show the security of our solution with respect to all properties stated above. The bank and the publisher are trusted to perform their function in the protocol correctly (i.e., the bank issues electronic payments and cashes its previously issued valid coins and the publisher announces the events correctly),

but we also design the protocols to be resilient to collision between different participants. For example, a payer, a payee, and the publisher should not be able to conspire to over-spend a bank-issued payment. Similarly, the bank, the publisher, and payees should not be able to conspire to link a coin to the identity of an honest payer.

As stated above, there are four players in the conditional e-payment system: the bank, the publisher, the payer, and the payee. We will assume that the bank will setup a public-private key pair prior to any payment can be made. The functionality of the system and the interaction between the bank, a payer, and a payee can be described using the following algorithms (the publisher does not interact with other parties, it has a passive role of announcing events and their outcomes):

Payment Generation: A protocol between the bank and a payer that allows the payer to obtain electronic payments from the bank. The bank withdraws from the payer's account the value of the electronic coin it issued.

Conditional Transfer: A protocol between a payer and a payee during which the payer transfers an electronic payment to the payee in such a way that the payee will be able to cash the payment only after the favorable outcome of the agreed-upon condition.

Validating the Payment: After the publisher announces the outcome of the event, if the outcome was favorable to the payee, the payee will use the information posted by the publisher to transform the conditional payment into a casheable coin.

Cashing the Payment: Cashing can be done by either the payee in case of favorable outcome of the condition or by the payer otherwise. In either case the claimant anonymously submits an electronic payment to the bank and receives cash in the amount of the payment.

Identifying Double-Spenders: This algorithm is invoked by the bank on input a coin's serial number s and two validity proofs for it. If a payee does not attempt to spend the same coin twice (with the same proof), this algorithm will reveal the identity of the payer (who either transferred the coin to more than one payee or cashed the coin that was also casheable by a payee).

The solution of [24] also included a **Payment Activation** protocol: a protocol between the bank and the payer, where the payer anonymously contacts the bank and activates the electronic payment issued during the payment generation protocol. This protocol, however, is not required.

We now give a more formal definition of a conditional e-payment system and its security properties.

Correctness. If an honest payer generates a conditional payment and engages in a conditional transfer protocol with a payee, then the payee will accept. If a payee is honest and accepts during the conditional transfer protocol, then in case of the favorable outcome the payee will be able to cash the payment. If a payer is honest and does not transfer the payment or transfers the payment and the outcome is unfavorable to the payee, the payer will be able to cash the payment.

Anonymity. When addressing anonymity of users, we need to consider two different cases: anonymity of payers and anonymity of payees. Anonymity of payers means that other participants (e.g., the bank, the publisher, and the payee) cannot link the coin an (honest) payer transfers to a payee to the payer's identity (assuming that the payer does not double-spend the coin). Thus, we model the adversary \mathcal{A} as colluding bank, payees, and the publisher. \mathcal{A} will be able to create the bank's private and public keys and engage in queries, where \mathcal{A} executes the payment generation protocol with various users id_i . Then \mathcal{A} engages in a challenge conditional transfer and the consecutive payment validation, where it is interacting either with a real user or a simulator with no access to any user information. The anonymity requirement for the payer is such that, for any adversary \mathcal{A} , given all information \mathcal{A} receives during payment generation, transfer, and validation, \mathcal{A} is unable to distinguish between a real payer and a simulator with more than negligible probability.

To model anonymity of payees, the adversary \mathcal{A} will represent the bank colluding with payers. In this case, \mathcal{A} will be able to create the bank's key, create users, and issue coins. To ensure that a payee's identity is not revealed at any point during the protocols, we require that \mathcal{A} cannot distinguish between a real user id_j and a simulator (with no access to user-specific information) with more than negligible probability during both the conditional transfer protocol and cashing payment protocol.

Balance. There are two parts to this property. From the bank's point of view, we would like to assure that no coalition of dishonest payers and payees can cash more coins than they withdrew. This is often shown by treating the payment generation protocol as a proof where the user plays the role of the prover and the bank plays the role of the verifier. Let x denote the user's input to the payment generation protocol. Then if the bank accepts at the end of the protocol, there will be a knowledge extractor that extracts a witness $w = s$, which is the serial number associated with the issued payment. Let the adversary \mathcal{A} engage in a number of payment generation, payment transfer, and payment cashing protocols. Let on i th successful execution of the payment generation protocol the extractor's output be (x_i, s_i) . We say that adversary wins if, for any number n of payment generation protocol executions, \mathcal{A} is able to cash a payment with a serial number $s \notin \{s_1, \dots, s_n\}$. We say that the balance property holds if \mathcal{A} has at most negligible probability of winning.

From the payer's point of view, if the payment was transferred to a payee and the outcome is unfavorable to the payee, that payee should not be able to produce a valid casheable payment. To be able to show this, we let \mathcal{A} act as either the payer or the payee with various users (by possibly corrupting other users) and, when acting as the payee, request favorable outcome for its events. Then \mathcal{A} , acting as the payee, engages in a challenge conditional transfer protocol with an uncorrupted payer, such that the outcome of this event is unfavorable. We say that the protocol is secure if any \mathcal{A} can recover a casheable token with at most negligible probability.

Double-spending. In this case the adversary \mathcal{A} represents a payer possibly in coalition with one or more payees; the bank is assumed to be honest. \mathcal{A} engages in the payment generation, conditional transfer, and cashing protocols as many times as it likes. \mathcal{A} wins the game if the bank accepts two requests to cash the same payment with serial number s without being able to recover the identity of \mathcal{A} . We say that identification of double-spenders is achieved if any \mathcal{A} can succeed at double-spending with at most negligible probability.

Deniability. Deniability was defined in [24] as the inability of a payer or a payee to prove to outside parties that he or she participated in a conditional payment protocol. This issue, however, requires further investigation to specify what constitutes a proof of engagement in such a protocol. If, for example, the format or the use of coins issued for a conditional payment protocol differs from coins issued by the bank for different purposes, then the mere fact that a user requests the bank to issue a conditional e-payment token to her indicates the intent to engage in a conditional transfer protocol. In this case, both the solution of Shi et al. and our solution fail to provide deniability. Furthermore, we argue that in many applications of conditional e-payments deniability might not be a requirement or a different type of deniability might be preferred (where, e.g., a participant – a payer or a payee – cannot provide a proof that its peer engaged in a conditional transfer protocol). Thus, we do not further treat deniability in this work.

3 Preliminaries

In this section, we review certain cryptographic primitives used as building blocks in our solution, as well as their security guarantees.

3.1 Zero-Knowledge Proofs of Knowledge

Prior literature provides efficient zero-knowledge proofs of knowledge (ZKPK) for a variety of statements, with many efficient proofs being based on the discrete logarithm problem (see, e.g., [14, 13, 11, 3, 4]). Camenisch and Stadler [12] introduced notation for various proofs of knowledge and we follow their notation here. For example,

$$PK\{(\alpha, \beta, \gamma) : A = g^\alpha h^\beta \wedge B = g^\alpha h^\gamma \wedge (\alpha \geq a)\}$$

denotes a ZKPK of integers α , β , and γ , where $A = g^\alpha h^\beta$, $B = g^\alpha h^\gamma$, and $\alpha \geq a$.

ZKPKs used in our protocols are proof of knowledge of the discrete logarithm representation, equality of discrete logarithms, and linear equations on the discrete logarithms, solutions to which are well known. We also utilize proof of knowledge that a discrete logarithm is the product of two other committed values [11].

3.2 Signature Schemes

In this work, we use signature schemes with protocols due to Camenisch and Lysyanskaya [8, 9]. These schemes have two protocols associated with them:

(i) they allow a user to obtain a signature on a committed value without revealing that value to the signer; and (ii) they enable the user to convince a third party that she possesses a signature on a certain message.

The commitment scheme used is the Pedersen commitment scheme [22]. Recall that in this scheme the public parameters are a group G_q of prime order q such that the discrete logarithm problem in G_q is hard and generators g_0, g_1, \dots, g_k . In order to compute a commitment to $x_1, \dots, x_\ell \in \mathbb{Z}_q$, we choose $r \in \mathbb{Z}_q$ at random and compute $\text{com}(x_1, \dots, x_\ell; r) = g_0^r \prod_{i=1}^{\ell} g_i^{x_i}$. This commitment is unconditionally hiding (i.e., $\text{com}(x_1, \dots, x_\ell; r)$ reveals no information about x_1, \dots, x_ℓ) and is computationally binding (assuming that the discrete logarithm problem is hard in G_q , the sender cannot open the commitment to values other than x_1, \dots, x_ℓ).

Then given a commitment $\text{com}(x_1, \dots, x_\ell; r)$, it is possible to obtain the signer's CL signature $\sigma(x_1, \dots, x_\ell)$ without revealing any information about the values x_1, \dots, x_ℓ to the signer. Furthermore, possession of $\sigma(x_1, \dots, x_\ell)$ allows its owner to use commitments to x_1, \dots, x_ℓ to prove to other parties that she has the signer's signature on the values included in the commitments without revealing additional information about the signed values themselves. If this protocol is combined with a ZK proof that the values included in these commitments satisfy certain properties, it becomes possible to convince a third party that the prover possesses a CL signature that meets these conditions without disclosing additional information about the signed values.

The signature scheme [8] relies on the Strong RSA assumption for its security. The scheme [9] relies on LRSW assumption in groups with bilinear maps. Our and other e-cash solutions built on such schemes require certain ZK proofs to be non-interactive, which is normally done by applying Fiat-Shamir heuristic [18]. Such non-interactive ZK proofs are, however, secure only in the random oracle model. Recent work by Belenkiy et al. [2] is the first to give new signatures with protocols, called P-signatures, that have non-interactive protocols without relying on the random oracle model. In particular, they utilize techniques of Groth and Sahai [19] to permit non-interactive zero-knowledge proofs that the contents of a commitment has been signed and that a pair of commitments are committed to the same value. The protocols used in this work, however, involve proofs of more general statements than equality, and more research is needed to determine what types of statements about discrete logarithms can be proven non-interactively using techniques of Groth and Sahai.

3.3 Verifiable Encryption

Verifiable encryption is a protocol between an encryptor and verifier that allows the encryptor to convince the verifier that encryption was performed correctly. Given a public encryption key pk and a commitment $C = \text{com}(x)$, this protocol allows the encryptor to produce encryption, $E_{pk}(x)$, of the opening of C , such that the verifier can accept an invalid encryption only with a negligible probability. Then given the corresponding decryption key sk and the protocol transcript for $E_{pk}(x)$, opening of C can be computed efficiently. Camenisch

and Damgård [5] provide techniques for converting any semantically secure encryption scheme into a verifiable encryption scheme, and this is what adds the logarithmic factor to the complexity of our scheme. For our purposes, any secure verifiable encryption scheme will suffice.

4 Conditional E-Payments

4.1 Description and Intuition

With the recent advances in e-cash systems, a natural place to seek alternatives for expensive cut-and-choose techniques of Shi et al. [24] conditional e-cash is to look at other e-cash solutions. Recent e-cash systems such as, e.g., [6,7,10] use CL-signatures with constant overhead as their building block. Using CL-signatures, it is possible to construct an e-cash system using a known method as follows: to generate a payment, a user obtains the bank's signature on (id, s, t) , where id is the user's identity, s is the serial number of the coin, t is the blinding value without the bank knowing s or t . Then when the user would like to spend the coin at a merchant, the user provides the merchant with commitments to id , s , and t and a ZK proof that she possesses a signature on these values. The merchant chooses a random value R (computed as a function of the merchant's identity and additional information provided by the merchant) and communicates R to the user. The user reveals to the merchant the serial number s and the result of evaluating the double-spending equation $D = id + R \cdot t \bmod q$ on R , as well as provides a ZK proof of correctness of both s and D (i.e., that s appears in the signature, and D is computed using id and t from the signature). Now, if only a single value D is computed for a coin, it does not reveal any information about the user's identity id (since t was chosen at random); but revealing two such values for different R 's reveals the identity of the user.

The biggest difference between the regular e-cash setting and conditional e-payments is that (i) in e-cash a coin transferred to a merchant carries a validity proof that is bound to the merchant (and only that merchant can cash it), while in conditional e-payments each payee is to stay anonymous, and (ii) the only way for a user to spend a coin in e-cash is through a merchant, while in conditional e-payments, the payer is able to cash coins herself. To address the first item, we can require each (anonymous) payee to challenge the payer on a randomly chosen value R , which, unlike the e-cash systems, is not bound to the payee's identity. This will ensure that if the payer transfers the coin to more than one payee, the payer's identity can be recovered. In case when the payer does not transfer the payment or the outcome of the event is unfavorable, the payer can transfer a coin to himself and cash it anonymously. This, however, creates a problem, as the payer will be able to quickly cash the coin using the payee's R before the payee learns the outcome of the condition. And when the (honest) payee attempts to cash the same payment, the request will be denied by the bank as duplicate (without the ability to uncover the payer's identity).

This could be fixed in the same way as in [24]: a trusted authority (e.g., the bank) keeps track of payment transfers¹. We, however, are interested in a solution that can be performed completely off-line, without the need of the bank to participate in payment transfers. Even though both participants are to stay anonymous, we let the payee to obtain a proof of the payment in an oblivious way, without the payer knowing the value of R . That is, the payee sends to the payer a commitment to R , g^R , and the payer then transfers that commitment into g^D and also provides a proof that the value was formed as prescribed. Note that knowledge of R is required to be able to cash the payment, and the payer is prevented from doing so. Given commitments to the double spending equation evaluated on different values, it is still possible to recover the identity of the payer (more precisely, now the bank will be able to recover g^{id} instead of the value of id itself, but this does not affect the security of the solution²).

In the above solution, the payee's participation in the conditional transfer and subsequent cashing protocols is linkable due to the use of the payer's validity proofs during cashing. Thus, to prevent colluding payer and the bank from recovering the payee's identity, the payee must stay anonymous during the deposit protocol. This means that either the payer cashes the e-payment anonymously or exchanges it for another anonymous token such that its generation and deposit protocols cannot be linked together (and this is what was suggested in [24]). Furthermore, the techniques we are utilizing do not seem to permit a payee's participation in the conditional transfer and cashing protocols to be unlinkable, as the payee is unable to modify the ZK proof generated by the payer. We leave this unlinkability issue as an open problem.

The final issue that remains to be addressed is the conditional nature of the transfer, where the payee's payment must remain uncasheable at the time of the transfer while still ensuring that the payee can verify all of the payer's proofs. We solve this by conditionally hiding only partial information which is necessary for cashing the payment. In particular, the payee will be able to obtain access to the serial number s only in case of favorable outcome of the condition.

We model events as follows: When event \mathcal{E} is announced, the publisher posts the public key $pk_{\mathcal{E}}$ associated with this event. Later, when the outcome of the event is determined, if the outcome satisfies the condition, the publisher releases the corresponding private key $sk_{\mathcal{E}}$. In case of unfavorable outcome, the publisher does not release any additional information. At the time of transfer, the payer encrypts the payment-related information using $pk_{\mathcal{E}}$, and the payee will be able

¹ In more detail, a coin consists of two halves (the right half is used for conditional transfers and the left half is used to cash unspent payments) such that spending both halves reveals the identity of the payer. The payer first (anonymously) activates a payment using its serial number s . During the transfer protocol, the payee contacts the bank with s and obtains a secret value that must be presented to the bank upon cashing the right half of the coin. Thus, the payer can cash only the left half.

² Furthermore, in previous e-cash schemes [6,7,10], each user was identified by a public key instead of her identity. Then for user U , the secret key is $sk_U \in \mathbb{Z}_q$ and the corresponding public key is $pk_U = g^{sk_U}$. Using our terminology, the secret key is id and the public key is g^{id} .

to decrypt it and produce a casheable coin only if the private key $sk_{\mathcal{E}}$ is announced. To ensure that the payer does not cheat during the encryption process, we employ verifiable encryption. More precisely, the payer forms a commitment to the serial number s and verifiably encrypts s using $pk_{\mathcal{E}}$. The payee will accept the payment only if she is able to verify the correctness of the encryption.

We also make use of another key pair $(pk_{\bar{\mathcal{E}}}, sk_{\bar{\mathcal{E}}})$ that correspond to the event $\bar{\mathcal{E}}$ opposite of \mathcal{E} (that is, $sk_{\bar{\mathcal{E}}}$ is published in case of unfavorable outcome of the event \mathcal{E}). The key $pk_{\bar{\mathcal{E}}}$ enables encryption of the value R to the payer, which he decrypts in case of unfavorable outcome and cashes the coin back.

4.2 The Scheme

Our scheme is as follows: The common parameters to all players consist of a group G of prime order q (chosen according to a security parameter 1^κ) and generators g_0, g_1, \dots . The bank generates its private signing key sk_B and the corresponding verification key pk_B using one of the CL-signature algorithms. This key will be used to sign e-payments the bank issues.

Payment Generation: The interaction between a payer and the bank is as follows:

1. The payer identifies herself as having id . The bank and the payer also agree on the amount of withdraw v . (We assume that $id, v \in \mathbb{Z}_q$.)
2. The payer with the help of the bank computes a commitment to s, t, id , and v , where s is a random value used as the payment's serial number (jointly generated by the payer and the bank). To achieve this:
 - (a) The payer chooses $s_P, t \in \mathbb{Z}_q$ at random, forms a commitment $C_1 = com(s_P, t; r) = g_0^r g_1^{s_P} g_2^t$, and sends it to the bank along with a ZK proof of knowledge of the representation of C_1 with respect to bases g_0, g_1 , and g_2 .
 - (b) The bank chooses $s_B \in \mathbb{Z}_q$ at random and sends it to the payer.
 - (c) Both the payer and the bank use s_B, id , and v to locally compute $C = com(s, t, id, v; r)$, where $s = s_P + s_B$. This is done by setting $C = C_1 \cdot g_1^{s_B} \cdot g_3^{id} \cdot g_4^v$.
3. The payer and the bank run a protocol for obtaining a signature on a committed value, at the end of which the payer has bank's signature $\sigma_B(s, t, id, v)$.
4. The bank debits the payer's account with amount v .
5. The payer stores $(id, s, t, v, \sigma_B(s, t, id, v))$ as his e-payment token.

When the payer would like to transfer the payment to a payee, both of them need to agree on the public condition \mathcal{E} announced by the publisher. The public key $pk_{\mathcal{E}}$ is then used during the conditional transfer.

In the conditional transfer protocol, the payer needs to convince the payee in the validity of the payment, which is done by using protocols associated with CL-signature and additional ZKPKs. Both participants stay anonymous during the protocol.

Conditional Transfer: The protocol between a payer with $(id, s, t, v, \sigma_B(s, t, id, v))$ and a payee proceeds in the following steps:

1. The payer forms commitments $C_1 = \text{com}(s; r_1)$, $C_2 = \text{com}(t; r_2)$, and $C_3 = \text{com}(id; r_3)$, and sends them along with v to the payee.
2. The payer sends to the payee a proof of knowledge π_1 of a CL signature from the bank on the openings of C_1 , C_2 , and C_3 as well as the value v .
3. The payee chooses $R \in \mathbb{Z}_q$ at random and sends $A = g^R$ to the payer.
4. The payer computes $B = g^{Rt+id}$, sends it to the payee, and executes a proof of knowledge π_2 of well-formedness of B . More precisely, the ZKPK is:

$$PK\{(\alpha, \beta, \gamma_1, \gamma_2) : B = A^\alpha g^\beta \wedge C_2 = g_1^\alpha g_0^{\gamma_1} \wedge C_3 = g_1^\beta g_0^{\gamma_2}\}$$

5. The payer verifiably encrypts s under the key $pk_\mathcal{E}$ using C_1 . The payee obtains $E_{pk_\mathcal{E}}(s)$, which includes the proof that the encryption was formed properly.
6. The payee verifiably encrypts R under the key $pk_\mathcal{E}$ using A . The payer obtains $E_{pk_\mathcal{E}}(R)$, which includes the proof that the encryption was formed properly.
7. As the result of this transfer, the payee stores $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_\mathcal{E}}(s))$ and the payer stores $(C_1, C_2, C_3, v, \pi_1, E_{pk_\mathcal{E}}(R), A, B, \pi_2, s)$.

After the outcome of the event is announced, if it was favorable to the payee, the payee obtains the private key $sk_\mathcal{E}$ corresponding to $pk_\mathcal{E}$ and performs the following payment validation procedure.

Validating the Payment: On input $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_\mathcal{E}}(s))$ and $sk_\mathcal{E}$, the payee decrypts $E_{pk_\mathcal{E}}(s)$ obtaining s and stores $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ as a casheable payment.

In case of unfavorable outcome, the payer transfers his coin into a casheable payment using $sk_{\bar{\mathcal{E}}}$ (i.e., given $(C_1, C_2, C_3, v, \pi_1, E_{pk_\mathcal{E}}(R), A, B, \pi_2, s)$, the payer decrypts $E_{pk_\mathcal{E}}(R)$ to obtain $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$. In case the payer did not engage in any conditional transfer protocol and would like to cash the coin back, the payer engages in the transfer protocol with himself without encrypting the serial number. In this case, the casheable coin is the same as in other cases.

The following protocol is performed anonymously.

Cashing the Payment: The claimant's input consists of a casheable coin $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$.

1. The claimant submits the payment $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ to the bank.
2. The bank verifies all proofs and, in particular, that $g^R = A$. The bank also checks whether the pair (s, R) was previously submitted to the bank. If the proofs are correct and (s, R) is fresh, the bank credits the claimant's account with amount v ; otherwise, it rejects the payment.
3. The bank records the payment $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ in the database of spent payments and searches for another record with s . If s has been used before, it invokes the identification algorithm.

Identifying Double-Spenders: If a coin with a serial number s can be found in the bank's database more than once, the bank identifies the guilty payer as follows:

- The bank locates two records (s, R_1, B_1) and (s, R_2, B_2) . (Recall that each $B_i = g^{R_1 t + id}$ for some value t unknown to the bank.)
- It computes the identity of the payer by first computing $d_1 = (B_1/B_2)^{(R_1-R_2)^{-1}}$ and then $d_2 = B_1/(d_1^{R_1})$. The value of d_2 will correspond to g^{id} , where id is the identity of the double-spender.
- The bank searches its database for an id that matches g^{id} .

We briefly show that d_2 indeed corresponds to the value g^{id} :

$$\begin{aligned} d_2 &= \frac{B_1}{d_1^{R_1}} = B_1 \left(\frac{B_2}{B_1} \right)^{(R_1-R_2)^{-1}R_1} = g^{R_1 t + id} \left(g^{R_2 t + id - R_1 t - id} \right)^{(R_1-R_2)^{-1}R_1} \\ &= g^{R_1 t + id} \left(g^{-t(R_1-R_2)} \right)^{(R_1-R_2)^{-1}R_1} = g^{R_1 t + id} g^{-R_1 t} = g^{id} \end{aligned}$$

Efficiency. Computation and communication in all of our protocols are constant (more precisely, linear in the security parameter), with the exception of verifiable encryption used in the transfer protocol. To achieve probability of cheating 2^{-k} , $O(k)$ encryptions must be performed during the transfer protocol. Additionally, to decrypt a value during payment validation, $O(k)$ decryptions are to be performed.

4.3 Security Analysis

Theorem 1. *Assuming the security of the CL-signature scheme and the verifiable encryption scheme, the above scheme is a secure conditional e-payment scheme in the random oracle model.*

Proof. We analyze the scheme w.r.t. the properties given in section [2](#).

Correctness. This property can be shown by examination.

Balance. We first show that dishonest users cannot cash more coins than what they withdrew. Let X_{pg} denote a knowledge extractor in the payment generation protocol that acts as the bank and X_{pk} denote a knowledge extractor for the proof of knowledge executed in step 2(a) of the protocol. Then during each execution of the payment generation protocol by the adversary \mathcal{A} , X_{pg} executes the protocol as the bank would with the exception that it also executes X_{pk} to obtain values id, s, t, v . After n executions of the protocol, X_{pg} has the knowledge of n serial numbers s_1, \dots, s_n . Now note that the soundness property of the proofs of knowledge prevents \mathcal{A} from successfully producing a valid serial number $s' \notin \{s_1, \dots, s_n\}$. Therefore, \mathcal{A} will attempt to deposit a coin for which it cannot honestly generate a valid proof. For \mathcal{A} to succeed in convincing the bank that it has a valid coin with serial number s' , \mathcal{A} must produce a proof that either \mathcal{A} knows the bank's signature on openings of C_1, C_2 , and C_3 or B is well-formed. The former can happen only with a negligible probability assuming that the

CL-signatures are secure, and the latter can also happen only with a negligible probability assuming that the discrete logarithm problem is hard. Therefore, \mathcal{A} can succeed in winning this game with at most negligible probability.

To show that the adversary \mathcal{A} , acting as a payee, is unable to spend its coin in case of unfavorable outcome of the event, let c_1, \dots, c_n denote the sequence of casheable coins \mathcal{A} acquires in the first part of the game (acting as both payers and payees), where the pairs $(E_{pk_{\varepsilon_1}}(s_1), s_1), \dots, (E_{pk_{\varepsilon_n}}(s_n), s_n)$ are the corresponding encryptions of coin serial numbers produced during the conditional transfer protocols and their decryptions. \mathcal{A} then engages in a challenge transfer protocol with an honest payer and obtains coin $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_{\varepsilon}}(s))$. Now assume that \mathcal{A} is able to recover a casheable payment from this coin without access to sk_{ε} . Then \mathcal{A} must either recover the correct value of s or produce a valid coin using another serial number s' . As was argued above, the latter is possible only with a negligible probability. \mathcal{A} will then attempt to recover s from the commitment C_1 and the associated proof of knowledge or encryption $E_{pk_{\varepsilon}}(s)$. Since the commitment is unconditionally hiding and the proof of knowledge is zero-knowledge, \mathcal{A} must use $E_{pk_{\varepsilon}}(s)$. Finally, assuming the security of the verifiable encryption scheme, \mathcal{A} can recover s from $E_{pk_{\varepsilon}}(s)$ with at most negligible probability. Therefore, \mathcal{A} can succeed in producing a casheable coin in this game with at most negligible probability.

Anonymity. Anonymity of a payee is trivially achieved, since at no point in the protocol the payee is required to present her identity or include it in the information exchanged between the payee and the payer or the bank.

To show payer anonymity, let the game setup be as described in section 2. Adversary \mathcal{A} represents other participants colluding together, who can create users and engages in payment generation protocols. During the challenge, \mathcal{A} is asked to engage in a conditional transfer protocol and execute the consecutive payment validation procedure with either a real user id_j or a simulator S without access to user id_j 's information. Our simulator S participates in a conditional transfer protocol by performing the following steps:

1. S chooses $id, s, t, v \in \mathbb{Z}_q$ and produces commitments $C_1 = com(s; r_1)$, $C_2 = com(t; r_2)$, and $C_3 = com(id; r_3)$.
2. S produces a simulated proof of knowledge π_1 of a CL signature from the bank on the openings of C_1, C_2, C_3 , and the value v . This requires usage of the corresponding simulator of CL-signatures.
3. After obtaining $A = g^R$, S computes $B = g^{Rt+id}$ and produces a proof of knowledge π_2 of well-formedness of B .
4. S verifiably encrypts s under the key pk_{ε} using C_1 producing $E_{pk_{\varepsilon}}(s)$.

At the end of this interaction, \mathcal{A} obtains $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, E_{pk_{\varepsilon}}(s))$. After executing the challenge transfer protocol (with either a real user or a simulator), \mathcal{A} is given pk_{ε} and validates the coin obtaining $(C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$. We next argue that a casheable coin produced by a real payer is indistinguishable from a casheable coin produced by the simulator.

First, the payment generation protocol does not allow the bank to learn any information about the coin-specific values s and t . When a payer is issued a CL-signature $\sigma_B(s, t, id, v)$ by the bank, all values in it are information-theoretically hidden (i.e., the signature is issued on the values (s, t, id, v, r) for a random value r rather than (s, t, id, v) ; this r is obtained from the commitment that the payer submits and information-theoretically hides the values both in the commitment and in the signature). Therefore, the values s and t that S chooses are indistinguishable from those chosen by real users.

Other values produced by S in the transfer protocol are commitments C_1, C_2, C_3 (which perfectly hide the values), real proofs of knowledge π_2 (which therefore are indistinguishable from a payer's proofs), real verifiable encryption $E_{pk_\varepsilon}(s)$ which consequently is decrypted, and one simulated proof of knowledge π_1 . This simulated proof differs from a real proof of knowledge of a CL signature, but due to the security of CL-signatures, \mathcal{A} can distinguish between a real and simulated proofs only with a negligible probability.

Double spending. There are two different ways for \mathcal{A} (representing a coalition of dishonest users) to double-spend a coin with serial number s : (1) by transferring it to two different (honest) payees, both of whom are able to cash it later, or (2) by transferring it to a single (honest) payee, recovering the payee's challenge R , and cashing the coin while the payee has a casheable coin.

In the first case, suppose the bank accepts two casheable coins $c_1 = (C_1, C_2, C_3, v, \pi_1, R, A, B, \pi_2, s)$ and $c_2 = (C'_1, C'_2, C'_3, v, \pi'_1, R', A', B', \pi'_2, s)$ with the same serial number s . Since there are knowledge extractors for all proofs included in the payments, \mathcal{A} knows $\sigma_B(s, t, id, v)$ and $B = g^{id+Rt}$ (resp., $B' = g^{id+R't}$) (more precisely, these conditions can fail with at most negligible probability). Now, because R and R' can happen to be the same only with a small probability, the protocol for identifying double-spenders in this case will succeed in recovering the identity id .

Next, consider the case where \mathcal{A} transfers its payment to a single honest payee, and the payee is later able to obtain a casheable coin (i.e., the condition outcome is favorable). \mathcal{A} might attempt to produce a casheable coin using the payee's challenge R from the transcript of the conditional transfer protocol. \mathcal{A} in this case has $(C_1, C_2, C_3, v, \pi_1, E_{pk_\varepsilon}(R), A, B, \pi_2, s)$. To produce a casheable coin, \mathcal{A} will have to recover R from either $A = g^R$ or $E_{pk_\varepsilon}(R)$. However, in the first case \mathcal{A} will have to solve the discrete logarithm which, by our assumption, is infeasible, and in the first case to circumvent verifiable encryption scheme $E(\cdot)$, which is assumed to be secure. Therefore, in this case \mathcal{A} also succeeds with at most negligible probability.

5 Further Transfer of Coins

In this section, we sketch how a payee can further transfer a conditional payment to another payee. Solutions to achieve transferable e-cash, including off-line systems, have been previously proposed in the literature (see, e.g., [15,21,11,20,23]),

but to the best of our knowledge, no solution of the kind we propose (or even another solution based on CL-signatures) has previously appeared in the literature.

Now each payee who would like to be able to further transfer conditional payments will need to be registered with the bank to permit recovering her identity in case of double-spending. Note that a payee will be able to transfer a coin under the same condition as the one used in issuing the payment to that payee.³ To transfer a payment, each payee can use the bank's signature on the payee's identity and other information similar to the coins themselves. This, however, will require a signature per transfer since using it more than once reveals the identity of its owner. To enable a user to perform multiple transfers using the same credential from the bank, we modify the double-spending equation: the property we now desire is that using this credential on a unique serial number once will leave the user anonymous, but using it twice on the same serial number will allow the identity of the user to be uncovered. More precisely, evaluating this equation on a pair (s_1, R_1) and (s_2, R_2) , where $s_1 = s_2$ and $R_1 \neq R_2$ will lead to recovery of the identity, but observing the results of evaluating the equation on any number of values $(s_1, R_1), (s_2, R_2), \dots, (s_n, R_n)$ where all s_i 's are unique does not reveal information about the user. To achieve this, our idea is to have the double-spending equation in the same form $D_i = id + R_i \cdot t$, but make t dependent on s_i . This means that different values of s_i 's will result in different values of t and therefore different equations, but evaluating it on the same s and two values of R will permit recovery of id . The function $t = f(s)$ should be deterministic and such that, knowing s , computing t is difficult. For example, we can compute t using a pseudo-random function and a secret key u known to the user only. Then the user will obtain the bank's signature on a pair (id, u) without revealing u to the bank, and during a coin transfer compute $t = f_u(s)$ and $D = id + Rt$. The function f should be such that the user is able to convince the other protocol participant in zero-knowledge that both t and D were computed as prescribed (and according to the bank's signature on id, u).

User Registration: The interaction between a user and the bank is as follows:

1. The user identifies herself as having id and sends to the bank a commitment $C' = com(u; r)$ for a randomly chosen $u \in \mathbb{Z}_q$.
2. The user proves to the bank in zero-knowledge that she knows the representation of C' with respect to bases g_0 and g_1 .
3. Both the user and the bank use id to locally compute $C = com(u, id; r)$ by setting $C = C' \cdot g_2^{id}$.
4. The user and bank run a protocol for obtaining a signature on a committed value, at the end of which the user has bank's signatures $\sigma_B(u, id)$.
5. The user stores $(id, u, \sigma_B(u, id))$ as her payment transfer credential.

The payment generation protocol and the conditional transfer protocol between the payer and the first payee remain mostly unchanged. The only difference in

³ Since the payee does not have access to the serial number s , she cannot use other conditions for hiding it. Furthermore, even when the payment becomes casheable after successful outcome of the original event, that payee still will not be able to construct verifiable encryption of s using the (payer's) commitment C_1 .

the first conditional transfer protocol is that, in order for the payee to further transfer her coin, she will need to prove statements involving the coin’s serial number (to prove that the double-spending equation was constructed correctly). Therefore, we modify the commitment C_1 generated in the first step of the conditional transfer protocol to be $C_1 = com(s)$. This means that $com(s)$ is of the form g^s and does not unconditionally hide the value of s . The knowledge of s , however, is required for cashing the payment, and recovering it from g^s requires solving the discrete logarithm problem. Therefore, we relax the hiding property of the commitment to permit further transfer of payments.

Now suppose that a payee would like to transfer her conditional payment to another payee. This operation can be performed any number of times, and we denote the chain of payees associated with some payment s as P_1, P_2, \dots . When P_1 transfers her conditional payment to P_2 using coin $c = (C_1, C_2, C_3, v, \pi_1, R_0, A, B, \pi_2, E_{pk_\varepsilon}(s))$, the new challenge R_1 is computed as a one-way function of the previous value R_0 and randomness r_1 contributed by P_2 . That is, $R_1 = f(R_0, r_1)$ and the idea is to chain the sequence of challenges R_0, R_1, R_2, \dots , so that future values in the chain are unpredictable to earlier participants and P_i cannot completely control the value of R_i . For P_2 to convince P_1 that R_1 was computed correctly, the computation must be verifiable in zero-knowledge without disclosing R_1 or r_1 . Therefore, we can, for example, use the verifiable random function $f_k(x) = g^{1/(k+x)}$ of Dodis and Yampolskiy [17] (which is the most efficient construction) as $R_i = f(R_{i-1}, r_i) = g^{1/(r_i+R_{i-1})}$.

Going back to evaluating the double-spending equation on coin- and user-specific values, we have $t = f_u(s)$. The above DY function can also be used here to compute t . But since computation of this function would require access to the serial number s , which is not known at the time of the transfer, commitment $C_1 = g^s$ can be used instead. If all participants use an agreed-upon method of mapping any g^s to an element s' of the appropriate group (i.e., \mathbb{Z}_q^*), the computation will be of the form $t = f_u(s')$.

We next describe the protocol for transferring a payment from P_1 to P_2 . Further transfers from P_i to P_{i+1} are performed analogously, and each new transfer maintains information about all previous transfers. P_1 ’s input is the original coin $c = (C_1, C_2, C_3, v, \pi_1, R_0, A, B, \pi_2, E_{pk_\varepsilon}(s))$.

Additional Transfer:

1. P_2 chooses r_1 and sends $C = com(r_1; r')$ to P_1 . P_1 sends R_0 to P_2 .
2. P_2 computes $R_1 = f(R_0, r_1)$ and sends g^{R_1} and a proof of its well-formness to P_1 .
3. P_1 , who is in possession of $\sigma_B(u, id_{P_1})$, sends to P_2 commitments $C_1^{(1)} = com(u; r'_1)$, $C_2^{(1)} = com(id_{P_1}; r'_2)$ and proves possession of the bank’s signature on the openings of these commitments with proof $\pi_1^{(1)}$.
4. P_1 computes $t_1 = f(s', u)$ and $g^{D_1} = g^{id_{P_1} + R_1 t_1}$. P_1 sends to P_2 g^{D_1} and a proof $\pi_2^{(1)}$ of its well-formness.
5. P_1 transfers c to P_2 . P_2 verifies all of the proofs in c and that R_0 was used in c and pays P_1 . P_2 stores c and $c_1 = (r_1, R_1, C_1^{(1)}, C_2^{(1)}, g^{D_1}, \pi_1^{(1)}, \pi_2^{(1)})$.

Each additional transfer adds $c_i = (r_i, R_i, C_1^{(i)}, C_2^{(i)}, g^{D_i}, \pi_1^{(i)}, \pi_2^{(i)})$ to the coin. This information is used to recover the identity of P_i in case of double spending. Payment validation is performed as before, by decrypting the serial number s , but now only the last person in the chain is entitled to the payment.

During the cashing protocol, the bank will challenge the user on a newly chosen value of R computed in the same way as in the above additional transfer protocol. The user cashing the coin will evaluate the double spending equation $D = id + R \cdot f_u(s') \bmod q$ using her transfer credential $\sigma_B(u, id)$. This applies to every user, including the original payer (note that in payer's case, transfer is performed using the coin $\sigma_B(s, t, id)$), while the cashing protocol also requires from that user transfer credential $\sigma_B(u, id)$.

Cashing the Payment:

1. The banks sends to the claimant a randomly chosen value r .
2. The claimant presents a coin c and transfer values c_1, c_2, \dots, c_i for $i \geq 0$ (i.e., the claimant is participant P_{i+1} in the chain).
3. The bank first verifies all values and proofs in c and each c_i . It then retrieves the value R_i from c_i and computes $R_{i+1} = f(R_i, r)$.
4. Similar to the additional transfer protocol, the claimant proves possession of transfer credentials $\sigma_B(u, id_{P_{i+1}})$ and sends to the bank $g^{D_{i+1}}$ along with a proof of its well-formness.
5. The bank stores all values and issues to the claimant a payment for amount v recorded in c .

Double-spending now can happen in different ways (in all of the following cases double-spending occurs only if the coins in question become casheable): (1) as before, a payer can transfer his coin to more than one payee, (2) a payee can transfer her coin to more than one payee, or (3) a payee can transfer her coin to another payee and cash the coin herself. In case (1), the payer's identity is recovered as before using the serial number s and two different values for R_0 and B . The case of dishonest payees is handled as follows. Suppose payee P_i transferred c to P_{i+1} and P'_{i+1} (case (2) above), and now there are two chains $P_1, \dots, P_i, P_{i+1}, \dots, P_n$ and $P_1, \dots, P_i, P'_{i+1}, \dots, P'_m$. When P_n and P'_m cash the payments, the bank will be able to retrieve g^{D_i} and $g^{D'_i}$ computed using R_i and R'_i , respectively, and recover id_{P_i} in the same way. Case (3) above is handled in a similar way. The difference is that the bank obtains one value g^{D_i} that encodes id_{P_i} when user P_n cashes the payment. Another value $g^{D'_i}$ is obtained during the cashing protocol, where P_i is challenged on a new value R'_i .

6 Conclusions

Conditional e-payments, first introduced by Shi et al. [24], allow an electronic payment between two parties to be based on the outcome of a condition not known in advance. This work formalizes the security of a conditional e-payment scheme and gives a solution based on CL-signatures. Compared to work of [24],

our scheme completely avoids cut-and-choose techniques, thus exponentially reducing the protocols' overhead. We also eliminate the need for the bank to be involved in all conditional transfer protocols by making the protocol off-line.

Another significant contribution of this work is an extension that permits a conditional payment to be further transferred to other users. This extension builds a chain of users involved in a transfer of a particular coin, and each chain contains enough information to prevent double-spending by each user in the chain.

While our solutions satisfy the stated goals, there are a few directions that can be pursued in refining the properties we achieve. In particular, it would be interesting to see if the information used in the conditional payment protocol and the consecutive cashing protocols could be made unlinkable (this, however, is likely to require drastically different tools). Also, some of our constructions use commitments of the form g^x , which do not unconditionally hide x . Thus, it would be desirable to achieve stronger hiding properties.

References

1. Anand, R., Madhavan, C.: An online, transferable e-cash payment system. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 93–103. Springer, Heidelberg (2000)
2. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Non-interactive anonymous credentials. In: Theory of Cryptography Conference (TCC 2008). LNCS, vol. 4948. Springer, Heidelberg (2008)
3. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
4. Bresson, E., Stern, J.: Proofs of knowledge for non-monotone discrete-log formulae and applications. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 272–288. Springer, Heidelberg (2002)
5. Camenisch, J., Damgard, I.: Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000)
6. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
7. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Balancing accountability and privacy using e-cash. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, Springer, Heidelberg (2006)
8. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
9. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
10. Camenisch, J., Lysyanskaya, A., Meyerovich, M.: Endorsed e-cash. In: IEEE Symposium on Security and Privacy, pp. 101–115 (2007)

11. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
12. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
13. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report No. 260, ETH Zurich (1997)
14. Chaum, D., Evertse, J.-H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
15. Chaum, D., Pedersen, T.: Transferred cash grows in size. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 390–407. Springer, Heidelberg (1993)
16. Dingledine, R., Mathewson, N., Syverson, P.: TOR: The second generation onion router. In: USENIX Security Symposium (2004)
17. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
18. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
19. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology - EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
20. Jeong, I., Lee, D., Lim, J.: Efficient transferable cash with group signatures. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 462–474. Springer, Heidelberg (2001)
21. Okamoto, T., Ohta, K.: One-time zero-knowledge authentications and their applications to untraceable electronic cash. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E81-A(1), 2–10 (1998)
22. Pedersen, T.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
23. Saxena, A., Soh, B., Zantidis, D.: A digital cash protocol based on additive zero knowledge. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3482, pp. 672–680. Springer, Heidelberg (2005)
24. Shi, L., Carbutar, B., Sion, R.: Conditional e-cash. In: Financial Cryptography and Data Security (FC 2007). LNCS, vol. 4886, pp. 15–28. Springer, Heidelberg (2007)

Anonymity in Transferable E-cash^{*}

Sébastien Canard¹ and Aline Gouget²

¹ Orange Labs R&D, 42 rue des Coutures, BP6243, F-14066 Caen Cedex, France

² Gemalto, 6, rue de la Verrerie, F-92190 Meudon, France

Abstract. Regular cash systems provide both the *anonymity* of users and the *transferability* of coins. In this paper, we study the anonymity properties of transferable e-cash. We define two natural additional levels of anonymity directly related to transferability and not reached by existing schemes that we call *full anonymity (FA)* and *perfect anonymity (PA)*. We show that the FA property can be reached by providing a generic construction and that the PA's cannot. Next, we define two restricted perfect anonymity properties and we prove that it is possible to design a transferable e-cash scheme where a bounded adversary not playing the bank cannot recognize a coin he has already owned.

1 Introduction

Electronic cash systems aim at emulating regular cash. Users withdraw coins from a bank, and next pay merchants using them. Then, merchants deposit coins to the bank. Even if the property of *transferability* (i.e. received cash can be spend later without involving the bank) is seen as a fundamental property of regular cash, it is usually disregarded in the electronic setting. This lack of interest for transferable e-cash may be explained by the result given in [6] showing that it is impossible to transfer a coin without increasing its size. However, this apparent drawback is not always unacceptable for applications depending on the available amount of storage data. The main advantage of the transferability of e-cash would be the decrease of communications between the bank and all users.

The anonymity in (non transferable) e-cash systems is well-studied in the literature. When introducing the transferability property into e-cash schemes, new notions of anonymity appear that have not already been described in the literature. These new anonymity notions related to transferable e-cash are studied in this paper.

As far as we know, the first transferable e-cash schemes that provides a weak level of anonymity has been proposed in [10,11]. The anonymity level is said *weak* since the spenders identities are protected but it is possible to link several spends of the same user.

Another method for transferring e-cash has been presented in [14,6]. The anonymity level of this scheme is said *strong* since the spender identities are

* This work has been financially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by the French Agence Nationale de la Recherche and the TES Cluster under the PACE project.

protected and it is not possible to link several spends of the same user. Very recently, two transferable e-cash schemes have been proposed in [5]. Both schemes improve the efficiency of [14,6] by reducing the number of communications between the bank and users. One scheme offers a computational *strong* anonymity while the other one offers an unconditional *strong* anonymity. However, none of these schemes offers a “perfect” anonymity of spends since it is always possible for an adversary to recognize a coin that he has previously seen being spent.

There is a gap between the highest level of anonymity achieved by the transferable e-cash schemes of the state-of-the-art and the impossibility result given in [6] showing that transferable e-cash cannot fulfil an *unconditional* “perfect” anonymity since an unbounded payer can always recognize his own money if he sees it later in a payment.

In this paper, we contribute to reduce this gap, in one hand by showing the possibility for an e-cash system to fulfil higher levels of anonymity, and on the other hand by proving that a computational payer can always recognize his own money if he sees it later in a payment, meaning that transferable e-cash cannot provide a *computational* “perfect” anonymity.

In Section 2, we give formal definitions for transferable e-cash. In Section 3, we focus on the security properties related to anonymity and we introduce two new properties: the *Full Anonymity (FA)* meaning that the adversary \mathcal{A} is not able to recognize a coin he has already observed during a spending between honest users, and the *Perfect Anonymity (PA)* meaning that \mathcal{A} is not able to decide whether or not he has already owned a coin he is receiving. In Section 4, we show that a transferable e-cash scheme can fulfil the FA property by providing a generic construction, and that no scheme can fulfil the PA property by improving the impossibility result given in [6]. In Section 5, we give evidence that it is possible to design a PA scheme if we assume that the (not unbounded) adversary is not the bank. Finally, we conclude in Section 6.

2 Transferable E-cash

In this section, we define the algorithms of transferable e-cash, the variables and oracles used by the adversaries, and the classical security properties that are not related to anonymity; anonymity properties are defined in Section 3. Note that our model can easily be extended to wallets by using the compact e-cash techniques [2].

2.1 Algorithms

A transferable e-cash system involves two types of player: a bank \mathcal{B} and a user \mathcal{U} . A coin is represented by an identifier Id and some values π needed to prove its validity.

- $\text{ParamGen}(k)$ is a probabilistic algorithm that outputs the parameters of the system Par (including the security parameter k).
- $\text{BKeyGen}(Par)$ (resp. $\text{UKeyGen}(Par)$) is a probabilistic algorithm executed by \mathcal{B} (resp. \mathcal{U}) that outputs the key pair $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ (resp. $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$).

- **Withdraw**($\mathcal{B}(sk_{\mathcal{B}}, pk_{\mathcal{B}}, pk_{\mathcal{U}}, Par)$, $\mathcal{U}(sk_{\mathcal{U}}, pk_{\mathcal{U}}, pk_{\mathcal{B}}, Par)$) is an interactive protocol where \mathcal{U} withdraws from \mathcal{B} one coin. At the end, \mathcal{U} either gets a coin $C = (Id, \pi)$ and outputs *OK*, or outputs \perp . The output of \mathcal{B} is either its view $\mathcal{V}_{\mathcal{B}}^W$ of the protocol (including $pk_{\mathcal{U}}$), or \perp .
- **Spend** ($\mathcal{U}_1(Id, \pi, pk_{\mathcal{U}_2}, Par)$, $\mathcal{U}_2(sk_{\mathcal{U}_2}, pk_{\mathcal{B}}, Par)$) is an interactive protocol where \mathcal{U}_1 gives a coin to \mathcal{U}_2 . At the end, either \mathcal{U}_2 outputs a coin $C = (Id_C, \pi_C)$ or \perp , and either \mathcal{U}_1 saves that C is a spent coin and outputs *OK*, or \mathcal{U}_1 outputs \perp .
- **Deposit** ($\mathcal{U}(Id, \pi, sk_{\mathcal{U}}, pk_{\mathcal{U}}, pk_{\mathcal{B}}, Par)$, $\mathcal{B}(sk_{\mathcal{B}}, pk_{\mathcal{B}}, pk_{\mathcal{U}}, \mathcal{L}, Par)$) is an interactive protocol where \mathcal{U} deposits a coin (Id, π) at the bank \mathcal{B} . If (Id, π) is not consistent/fresh, then \mathcal{B} outputs \perp_1 . Else, if Id already belongs to the list of spent coins \mathcal{L} , then there is an entry (Id, π') and \mathcal{B} outputs (\perp_2, Id, π, π') . Else, \mathcal{B} adds (Id, π) to its list \mathcal{L} , credits \mathcal{U} 's account, and returns \mathcal{L} . \mathcal{U} 's output is *OK* or \perp .
- **Identify** (Id, π, π', Par) is a deterministic algorithm executed by \mathcal{B} that outputs a public key $pk_{\mathcal{U}}$ and a proof Π_G . If the users who had submitted π and π' are not malicious, then Π_G is evidence that $pk_{\mathcal{U}}$ is the registered public key of a user that double-spent a coin.
- **VerifyGuilt**($pk_{\mathcal{U}}, \Pi_G, Par$) is a deterministic algorithm that can be executed by any actor. It outputs 1 if Π_G is correct and 0 otherwise.

2.2 Global Variables

The set of user's public (resp. secret) keys is denoted by $\mathcal{PK} = \{(i, pk_i) : i \in \mathbb{N}\}$ (resp. $\mathcal{SK} = \{(i, sk_i) : i \in \mathbb{N}\}$; $sk_i = \perp$ if user i is corrupted). The set of views of supplied coin by oracles is denoted by \mathcal{SC} and the set of all coins owned by the oracles is denoted by \mathcal{OC} . The set of deposited electronic cash (corresponding to \mathcal{L}) is denoted by \mathcal{DC} .

2.3 Oracles

By convention, the name of an oracle corresponds to the action done by this oracle.

Creation and corruption of users. **Create**(i) executes $\mathbf{UKeyGen}(Par) = (sk_i, pk_i)$, it defines $\mathcal{PK}[i] = pk_i$ and $\mathcal{SK}[i] = sk_i$ and it outputs pk_i . The oracle **Corrupt**(i, pk_i) defines $\mathcal{PK}[i] = pk_i$ and $\mathcal{SK}[i] = \perp$ and it outputs *OK*. **Corrupt**(i) outputs the value $\mathcal{SK}[i] = sk_i$ and it updates $\mathcal{SK}[i] = \perp$.

Withdrawal protocol. **Suppl**() plays the bank side and it updates \mathcal{SC} by adding $\mathcal{V}_{\mathcal{B}}^W$ with bit 1 to flag it as a corrupted coin. The oracle **Withd**(U) plays the user \mathcal{U} side and it updates \mathcal{OC} by adding the value (U, Id, π) . The oracle **Withd&Suppl**(U) plays both sides and it updates \mathcal{OC} as for **Withd**(U) and \mathcal{SC} by adding $\mathcal{V}_{\mathcal{B}}^W$ with flag 0. It outputs the communications between \mathcal{B} and \mathcal{U} .

Spending protocol. The oracle $\text{Rcv}(U_2)$ plays the role of user \mathcal{U}_2 with secret keys of user U_2 and it updates the set \mathcal{OC} by adding a new entry (U_2, Id, π) . The oracle $\text{Spd}(U_1)$ plays the role of user \mathcal{U}_1 by spending a coin in \mathcal{OC} owned by user U_1 . It uses and updates the entry (U_1, Id, π) of \mathcal{OC} as the Spend protocol describes it.

The oracle $\text{Spd\&Rcv}(U_1, U_2)$ plays the role of both \mathcal{U}_1 and \mathcal{U}_2 and it executes the spending of a coin owned by user U_1 to user U_2 . It updates \mathcal{OC} by adding the value (U_2, Id, π) and by flagging the coin as spent by U_1 . It outputs all the communications of the spending.

Deposit protocol. $\text{CreditAccount}()$ plays the role of the bank and it updates the set \mathcal{DC} . If the executed Deposit protocol outputs (\perp_2, Id, π, π') , then the oracle CreditAccount runs the algorithm Identify and outputs the result of this algorithm on inputs (Id, π, π') . The oracle $\text{Depo}(U)$ plays the role of the user U during a Deposit protocol.

2.4 Classical Security Properties Not Related to Anonymity

Unforgeability. No collection of users can ever spend more coins than they withdrew.

Game. Let an adversary \mathcal{A} be a p.p.t. Turing Machine that has access to the set of all user's public keys \mathcal{PK} , the bank's public key pk_B and Par . \mathcal{A} can play as many times as he wants with the oracles: Create , Corrupt , Suppl , Withd\&Suppl , Spd , Spd\&Rcv , Rcv and CreditAccount .

Let q_W denote the number of successful queries to the oracle Suppl . Let q_R denote the number of successful queries to the oracle Spd . Let q_S be the number of successful queries to the oracle Rcv . The adversary \mathcal{A} wins the game if, at any time during the game, we have $q_W + q_R < q_S$.

Identification of double-spenders. No collection of users can double-spend a coin twice without revealing one of their identities.

Game. Let an adversary \mathcal{A} be a p.p.t. Turing Machine that has access to the \mathcal{PK} global variable, the bank's public key pk_B and Par . \mathcal{A} can play as many times as he wants with the oracles: Create , Corrupt , Suppl , Withd\&Suppl , Spd , Spd\&Rcv , Rcv and CreditAccount .

\mathcal{A} wins the game if, at any time of the game, the oracle CreditAccount outputs (\perp_2, Id, π, π') and the output of the oracle Identify on inputs (Id, π, π') is not a public key related to a secret key \perp in \mathcal{SK} .

Exculpability. The bank, even when cooperating with any collection of malicious users cannot falsely accuse users from having double-spent a coin.

Game. Let an adversary \mathcal{A} be a p.p.t. Turing Machine that has access to the \mathcal{PK} global variable, the bank's key pair (pk_B, sk_B) and Par . \mathcal{A} can play as many times as he wants with the oracles: Create , Corrupt , Withd , Spd , Spd\&Rcv , Rcv and Depo . At any time of the game, \mathcal{A} outputs two spends (Id_1, π_1) and (Id_2, π_2) . \mathcal{A} wins the game if the outputs of the algorithm Identify on inputs (Id_1, π_1, π_2) is a public key pk such that the related secret key in \mathcal{SK} is not \perp .

together with a valid proof Π_G , and the output of the algorithm `VerifyGuilt` on inputs (pk, Π_G) is 1.

3 Anonymity Properties in Transferable E-cash

In this section, we focus on security properties related to anonymity, remembering usual ones and introducing our two new ones.

3.1 Overview

The *Weak Anonymity (WA)* and the *Strong Anonymity (SA)* properties are classical properties. Informally speaking,

- an e-cash scheme fulfils the WA property if an adversary cannot link a spending to a withdrawal. However, the adversary may know if two spends are done by the same user or not.
- An e-cash scheme fulfils the SA property if it fulfils the WA property and if an adversary is not able to decide if two spends are done by the same user or not. However, the adversary may recognize a coin that he has already observed during previous spends.

We introduce two new anonymity properties directly related to the transferability property in e-cash that we call *Full Anonymity (FA)* and *Perfect Anonymity (PA)*. Informally speaking,

- an e-cash scheme fulfils the FA property if it fulfils the SA property and if an adversary is not able to recognize a coin that he has already observed during a spending between two honest users. However, the adversary may be able to recognize a coin he has already owned.
- An e-cash scheme fulfils the PA property if it fulfils the FA property and if an adversary is not able to decide whether or not he has already owned a coin he is receiving.

3.2 Description of the Game

Before defining the game, we need to recall that a transferred cash necessary grows in size \mathbb{G} . This property may be exploited by the adversary \mathcal{A} to win the game and break the anonymity property. Indeed, \mathcal{A} may choose two users that do not own coins of the same size so as to distinguish which one is used at the end of the game.

Consequently, in the following game, we impose that the two challenged users i_0 and i_1 own coins of the same size and the coin used during the final call to the `Spd` oracle should be one of these coins.

Game. Let an adversary \mathcal{A} be a p.p.t. Turing Machine that has access to the set of all user's public keys \mathcal{PK} .

1. \mathcal{A} is given $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$, Par and \mathcal{A} can play with the oracles: **Create**, **Corrupt**, **Withd**, **Spd**, **Rcv**, **Spd&Rcv** and **Depo**.
2. At any time, \mathcal{A} chooses two public keys $pk_{i_0}, pk_{i_1} \in \mathcal{PK}$, such that:
 - (a) $\mathcal{SK}[i_0] \neq \perp$ and $\mathcal{SK}[i_1] \neq \perp$;
 - (b) users i_0 and i_1 own coins of the same size;
 - (c) users i_0 and i_1 have been used only by a set of *authorized oracles* that depends on the power of the adversary \mathcal{A} (see below).
3. A bit b is secretly and randomly chosen and \mathcal{A} plays with $\text{Spd}(i_b)$.
4. \mathcal{A} outputs a bit b' .

3.3 Security Properties Related to Anonymity

We define four adversaries related to the four anonymity properties (WA, SA, FA and PA) that can be used to play the game described at Section 3.2. Indeed, the adversary is allowed to observe the transactions involving the coin that will be spend at step 3 with some specific restrictions depending on the anonymity property.

Definition 1 (Adversaries). We denote by i_0 and i_1 the two users chosen by the adversary at Step 2 of the game described in Section 3.2.

- The adversary \mathcal{A}_{WA} is allowed to manipulate i_0 and i_1 only with the oracles: **Create**, **Withd** and **Depo**.
- The adversary \mathcal{A}_{SA} is allowed to manipulate i_0 and i_1 only with the oracles: **Create**, **Withd**, **Spd**, **Spd&Rcv** with the additional restriction that i_0 and i_1 do not play the role of \mathcal{U}_2 and **Depo**.
- The adversary \mathcal{A}_{FA} is allowed to manipulate i_0 and i_1 only with the oracles: **Create**, **Withd**, **Spd**, **Spd&Rcv** and **Depo**.
- The adversary \mathcal{A}_{PA} is allowed to manipulate i_0 and i_1 with all the oracles except the **Corrupt** oracle.

Definition 2 (Anonymity properties). A transferable e-cash system fulfils the property $\mathcal{P} \in \{\text{WA}, \text{SA}, \text{FA}, \text{PA}\}$ if for an adversary $\mathcal{A}_{\mathcal{P}}$ playing the game described at Section 3.2, the probability that $b = b'$ differs from $1/2$ by a fraction that is at most negligible.

Remark 1. By construction, the anonymity properties are (exclusively) related one to the other as follows: $\text{PA} \Rightarrow \text{FA} \Rightarrow \text{SA} \Rightarrow \text{WA}$.

4 Study of Anonymity Properties

The schemes proposed in [14,6,5] fulfil both the WA and the SA properties but, as far as we know, the FA property (and consequently the PA property) is not achieved by any state of the art scheme. In this section, we first show that the FA property can be reached by providing a generic construction. Next, we prove that the PA property cannot be achieved.

4.1 Achieving the Full Anonymity Property

The difference between the SA game and the FA game is that the coin received at the challenge step by the adversary may have already been observed by \mathcal{A}_{FA} during a call to the Spd\&Rcv oracle, whereas this case cannot happen during the SA game. The generic construction we propose is built upon an SA scheme. The key idea of our construction is to modify an SA scheme to get an FA scheme by protecting the communications of the spending protocol using the establishment of a unilateral authenticated secure channel between the receiver and the spender in order to prevent an active adversary to recognize a coin that he has previously seen being spent.

We assume that \mathcal{S} is a transferable e-cash scheme that fulfils the SA property. From \mathcal{S} , we construct a transferable e-cash scheme \mathcal{S}' that fulfils the FA property. We re-define only the spending protocol of \mathcal{S} and we additionally use two building blocks: a secure symmetric encryption scheme $\mathcal{E} = (\text{Enc}, \text{Dec})$, and a unilateral authenticated key agreement protocol KE between two users (including a key-confirmation step) secure against an active adversary. In particular, KE is resistant to man-in-the-middle attacks.

A user \mathcal{U}_1 spends a coin to user \mathcal{U}_2 by first playing the KE protocol. At the end of the KE protocol, \mathcal{U}_1 and \mathcal{U}_2 share a unilateral authenticated session key K . Next, \mathcal{U}_1 and \mathcal{U}_2 play the Spend protocol of \mathcal{S} by encrypting all the communications using the algorithms Enc and Dec with the common session key K .

Theorem 1. *Under the assumptions that \mathcal{S} fulfils the SA property, and EK and \mathcal{E} are secure, the system \mathcal{S}' fulfils the FA property.*

Sketch of Proof. Assume that \mathcal{A}_{FA} breaks the FA property by determining between users i_0 and i_1 , which one is i_b . By definition \mathcal{A}_{FA} is not allowed to manipulate i_0 and i_1 with the oracle Rcv and thus \mathcal{A}_{FA} owns the coin of the challenge only at step 3 of the game. \mathcal{A}_{FA} should have seen it during the withdrawal of this coin (using the oracle $\text{With}(U)$) and possibly during spends between honest users (using the oracle Spd\&Rcv).

By assumption (\mathcal{S} fulfils the SA property), \mathcal{A}_{FA} cannot get the serial number of a coin involved in a withdrawal protocol. By construction of \mathcal{S}' , all communications related to the spending of a coin are encrypted with a unilateral authenticated ephemeral session key, which includes the communications related to a call to the oracle Spd\&Rcv . Thus, \mathcal{A}_{FA} has no information about the identifier of the coin embedded into the spending (\mathcal{A}_{FA} may know the entry number of the coin in \mathcal{OC} but not the serial number), except if \mathcal{A}_{FA} has succeeded in breaking either the security of KE to obtain the session key K or the security of \mathcal{E} to decrypt the communications without knowing the decryption key. \square

4.2 Impossibility of the Perfect Anonymity Property

It is known that a payer with unlimited computing power can always recognize his own money if he sees it later being spent [6], and thus the PA property cannot be achieved by an unlimited powerful adversary. In this section, we show that a

bounded adversary \mathcal{A}_{PA} , acting as the bank, can also win the anonymity game, meaning that the PA property cannot be achieved by transferable e-cash.

Attack against the PA Property. During the PA game, \mathcal{A}_{PA} creates users and corrupts some of them. At any time, \mathcal{A}_{PA} owns a set of valid coins $\{C_0, \dots, C_l\}$ that he got from his interactions with the oracle Spd .

\mathcal{A}_{PA} chooses two honest users i_0 and i_1 such that they have no coins. Next \mathcal{A}_{PA} chooses two coins C_0 and C_1 , spends coin C_0 to user i_0 and coin C_1 to user i_1 using the oracle Rcv . Then, \mathcal{A}_{PA} outputs i_0 and i_1 , according to the PA game. The challenger next chooses at random a bit b and \mathcal{A}_{PA} plays a Spend protocol with the oracle Spd on input the user i_b . Acting as the bank, \mathcal{A}_{PA} then simply executes the Deposit algorithm for the coins C_b and C_0 ¹. If a double spending is detected, then \mathcal{A}_{PA} outputs $b' = 0$ and he outputs $b' = 1$ otherwise. Thus, \mathcal{A}_{PA} can always succeed in guessing b .

5 Variants of the Perfect Anonymity Property

The attack described in Section 4.2 shows that the PA property cannot be achieved. In this section, we describe the two most natural ways to modify the PA game in order to prevent this attack. We define two distinct properties called PA_1^* and PA_2^* and show that both properties can be achieved. Next, we prove that there is no inclusion relation between the FA property and PA_1^* (resp. PA_2^*).

5.1 Additional Anonymity Properties

The first possibility to make impossible the attack described in Section 4.2 is to prevent the adversary from receiving the coin C_b at Step 3 of the game described in Section 3.2. Then the coin C_b may have been manipulated by the adversary before Step 3 but the adversary only observes the spending of coin C_b between two honest users at Step 3.

Definition 3 (Adversary PA_1^*). *The adversary $\mathcal{A}_{\text{PA}_1^*}$ can manipulate the challenged users with all the oracles except the Corrupt oracle.*

Game of the PA_1^* property. We modify the game described in Section 3.2 as follows. The steps 1 and 2 are unchanged. In step 3, the call to $\text{Spd}(i_b)$ is replaced by a call to $\text{Spd}\&\text{Rcv}(i_b, i)$ where i is a randomly chosen honest user.

The second possibility to avoid the attack against the PA property is to prevent the attacker to execute the deposit. We thus separate the power of the bank into two entities with distinct keys: \mathcal{B}_W (resp. \mathcal{B}_D) is responsible of the withdraw (resp. the deposit) part. In the new property, the adversary is not allowed to control \mathcal{B}_D . Moreover, the Deposit and Identify protocols should be protected by a secret key of the bank \mathcal{B}_D . We should prevent the adversary from being able to simulate the deposit phase, as a user can do when this phase is based on public algorithms.

¹ Even if this is a fraud, \mathcal{A}_{PA} can deposit the coin C_0 he has already spent.

Definition 4 (Adversary $\mathcal{A}_{\text{PA}_2^*}$). The adversary $\mathcal{A}_{\text{PA}_2^*}$ can manipulate the challenged users with all the oracles except the oracles `CreditAccount` and `Corrupt`.

Game of the PA_2^* property. We modify the game described in Section 3.2 as follows. Only Step 1 is modified: `Withd` is replaced by `Suppl`, and `Depo` is replaced by `CreditAccount`.

Remark 2. Note that the discussion on public [13] or secret [2] `Deposit` and `Identify` is controversial in non-transferable e-cash definitions.

5.2 Studying the PA_1^* Property

We first show that the PA_1^* property can be achieved by proving that the scheme \mathcal{S}' described in Section 4.1 is PA_1^* .

Theorem 2. The scheme \mathcal{S}' described in Section 4.1 is PA_1^* .

Sketch of Proof. Assume that $\mathcal{A}_{\text{PA}_1^*}$ breaks the PA_1^* property of \mathcal{S}' by determining between users i_0 and i_1 which one is i_b . Before Step 3 of the game, $\mathcal{A}_{\text{PA}_1^*}$ has observed the withdrawal of the coin of the challenge but cannot get the related serial number from it (\mathcal{S} fulfils the SA property). Moreover, $\mathcal{A}_{\text{PA}_1^*}$ may have owned many times the coin of the challenge using the oracles `Spd`, `Rcv` and `Spd&Rcv`. But, by construction of \mathcal{S}' , all communications related to the spending of a coin are encrypted with a unilateral authenticated ephemeral session key. Thus, all communications of the final call to the `Spd&Rcv` oracle are encrypted, which means that $\mathcal{A}_{\text{PA}_1^*}$ cannot recognize the serial number of the spent coin embedded into the spending, except if $\mathcal{A}_{\text{PA}_1^*}$ has succeeded in breaking either the security of KE or the security of \mathcal{E} by decrypting the communications without knowing the decryption key. \square

We then show that the previously defined anonymity properties and PA_1^* are independent (i.e. one property does not imply the other).

Proposition 1. WA (resp. SA, resp. FA) and PA_1^* are independent properties.

Proof. $\text{WA} \not\Rightarrow \text{PA}_1^*$. The scheme proposed by Okamoto and Ohta [10,11] fulfils the WA property but not the PA_1^* one since the serial number of a coin is not protected and it is not modified from one spend to another.

$\text{PA}_1^* \not\Rightarrow \text{WA}$. If we apply the general construction of Section 4.1 onto a transferable e-cash scheme that does not fulfil the WA property, we can easily show that the new scheme fulfils PA_1^* but not WA.

$\text{SA} \not\Rightarrow \text{PA}_1^*$. The schemes proposed in [14,6,5] fulfil the SA property but not the PA_1^* one since the serial number of a coin is not protected and it does not change from one spend to another.

$\text{PA}_1^* \not\Rightarrow \text{SA}$. From $\text{SA} \Rightarrow \text{WA}$ and $\text{PA}_1^* \not\Rightarrow \text{WA}$, we have $\text{PA}_1^* \not\Rightarrow \text{SA}$.

$\text{FA} \not\Rightarrow \text{PA}_1^*$. See Appendix A.

$\text{PA}_1^* \not\Rightarrow \text{FA}$. This is due to $\text{FA} \Rightarrow \text{SA}$ and $\text{PA}_1^* \not\Rightarrow \text{SA}$. \square

5.3 General Description of a PA_2^* Scheme

In this section, we want to prove that PA_2^* can be reached by a transferable e-cash scheme and the efficiency of the constructed scheme is out of the scope of this paper. The construction of a PA_2^* transferable e-cash scheme is less straightforward than the PA_1^* 's one. Indeed, we need to use an additional tool called *metaproof system* that has been introduced in [12] by de Santis and Yung.

Metaproofs. Roughly speaking, the metaproof tool corresponds to a NIZK (Non-Interactive Zero-Knowledge) proof of the existence of a NIZK proof to a statement. More precisely, they provide a metaproof system for 3SAT and prove that their metaproof system is a bounded NIZK proof system [1]. The metaproof system gives an *indirect* proof covered by additional encryption mechanism such that the metaprover possesses a zero-knowledge witness and does not necessary know the witness of the proof itself. Eventually, the metaproof can be applied recursively.

Overview of our PA_2^* transferable e-cash scheme. A spent coin is classically represented by at least a serial number S , a security tag T (that permits the identification of double-spenders) and a proof that S and T are correct. Then, a transferable spent coin should consists in at least a serial number S , a set of security tags $\mathcal{T} = \{T_1, \dots, T_l\}$ and a proof of validity.

We first notice that, in a PA_2^* e-cash scheme, a coin should be transferred without revealing any information on previous spends, even for the user that is receiving the coin. Moreover, the bank needs to retrieve the serial number and all security tags describing the history of the coin, which can be done by encrypting these values using the bank's public key.

Since a user should not be able to recognize a coin previously owned, the receiver should be able to verify the validity of the coin without being able to retrieve neither the serial number nor any security tag. Moreover, since the spender can next become a receiver of this coin, the spent coin should be modified at each spend so that she cannot recognize it. We thus need a cryptographic tool permitting someone to send the serial number, security tags and proofs without revealing nor knowing them but proving that they are valid, which can be done using metaproofs [12].

More precisely, if a user withdraws a coin, she spends it by computing the serial number S of the coin and the security tag T_1 , plus a proof of validity V_1 that S and T_1 are well-formed. If the receiver wants to spend this coin, she computes a security tag T_2 , she encrypts S and T_1 and she proves that T_2 is well-formed and that she knows the encryption of the serial number S , the encryption of the first security tag T_1 and a proof of validity V_1 without revealing the encrypted values nor V_1 , using in particular a metaproof.

Description of our PA_2^* Scheme. We largely use the proposal of transferable e-cash scheme from Canard, Gouget and Traoré [5] to describe our PA_2^* scheme, with the restriction that a user withdraws one coin at a time, and not a wallet. In the following, we only give a high level description of our scheme and we refer to [5] and [12] for details.

Setup. Let \mathcal{G} be a group of prime order p and g, g_0 be two random generators in \mathcal{G} . These data constitute the public parameters Par . Let \mathcal{H} be a cryptographic hash function. In the **BKeyGen** algorithm, \mathcal{B} computes two key pairs $(sk_{\mathcal{B},1}, pk_{\mathcal{B},1})$ and $(sk_{\mathcal{B},2}, pk_{\mathcal{B},2})$ of a CL signature scheme [3] that permit it to sign coins and enroll users, respectively. During the **UKeyGen** algorithm, each user \mathcal{U}_i obtains a CL (verifiable) signature $C_i = \text{Sign}(u_i, w_i)$ associated to his public key $pk_{\mathcal{U}_i} = g_0^{u_i}$ and a random data w_i . Let $\text{Enc}_{\mathcal{B}}$ be a secure verifiable probabilistic encryption scheme (e.g. El Gamal) to encrypt messages for the bank.

Withdrawal protocol. Following [2,5], a coin C withdrawn at the bank is a CL signature σ under the bank's public key $pk_{\mathcal{B},1}$ on the set of values (s, u_i, t, x) where u_i is the user secret key, s, t and x are random values; $C = (s, (u_i, t, x, \sigma))$. The value s implicitly defines the *serial number* of the coin and the value t implicitly defines the corresponding *security tag* (using the Dodis-Yampolskiy Pseudo Random Function [7]).

Spending of a withdrawn coin. A user \mathcal{U}_i , owning a coin $C = (s, (u_i, t, x, \sigma))$ withdrawn from \mathcal{B} , wants to spend a coin to a user \mathcal{U}_j .

1. \mathcal{U}_j computes $r_j = g_0^{\frac{1}{u_j + d_j}}$ where d_j represents some data related to the transaction. Next, \mathcal{U}_j sends r_j and d_j to \mathcal{U}_i .
2. \mathcal{U}_i computes $S = g^s$, $T_i = pk_{\mathcal{U}_i} g^{\frac{r_j}{t + d_j}}$ and a NIZK proof V_i that
 - \mathcal{U}_i knows a signature σ on s, u_i, t and x ,
 - $S = g^s$ and $T_i = pk_{\mathcal{U}_i} g^{\frac{r_j}{t + d_j}} = g_0^{u_i} g^{\frac{r_j}{t + d_j}}$,
without revealing s, t, u_i, x nor σ .
3. The spent coin is represented by $(S, \pi = (T_i, V_i, r_j, d_j))$.

First transfer of a coin. Let us now consider the user \mathcal{U}_j that has received a coin $(S, \pi = (T_i, V_i, r_j, d_j))$ from user \mathcal{U}_i during the above protocol. If \mathcal{U}_j wants to spend this coin to a user \mathcal{U}_k , he has to proceed as follows.

1. \mathcal{U}_k computes $r_k = g_0^{\frac{1}{u_k + d_k}}$ where d_k represents some data related to the transaction. Next, \mathcal{U}_k sends r_k and d_k to \mathcal{U}_j .
2. \mathcal{U}_j computes $T_j = pk_{\mathcal{U}_j} g^{\frac{r_k}{u_j + S + d_k}}$, $dS = \text{Enc}_{\mathcal{B}}(S)$, $dT_i = \text{Enc}_{\mathcal{B}}(T_i)$ and a NIZK proof V_j that
 - \mathcal{U}_j knows a signature C_j on u_j and w_j
 - $T_j = pk_{\mathcal{U}_j} g^{\frac{r_k}{u_j + S + d_k}} = g_0^{u_j} g^{\frac{r_k}{u_j + S + d_k}}$ and $r_j = g_0^{\frac{1}{u_j + d_j}}$,
 - dS and dT_i are correct encryptions of the unrevealed values S and T_i , respectively,
 - there exists a NIZK proof V_i proving that S and T_i are well-formed, using in particular r_j and d_j , and linked to a valid signature of the bank (this step corresponds to a metaproof as described in [12]),
without revealing $u_j, S, C_j, w_j, r_j, d_j, T_i$ nor V_i .
3. The spent coin is represented by $(dS, \pi = (T_j, dT_i, V_j, r_k, d_k))$.

Second transfer of a coin. Let us now consider the user \mathcal{U}_k that has received a coin $(dS, \pi = (T_j, dT_i, V_j, r_k, d_k))$ from user \mathcal{U}_j during the above protocol. If \mathcal{U}_k wants to spend it to \mathcal{U}_l , he has to proceed as follows.

1. \mathcal{U}_l computes $r_l = g_0^{\frac{1}{u_l+d_l}}$ where d_l represents some data related to the transaction. Next, \mathcal{U}_l sends r_l, d_l to \mathcal{U}_k .
2. \mathcal{U}_k computes $T_k = pk_{\mathcal{U}_k} g^{\frac{r_l}{u_k+dS+d_l}}$, $d^2S = \text{Enc}_{\mathcal{B}}(dS)$, $dT_j = \text{Enc}_{\mathcal{B}}(T_j)$, and $d^2T_i = \text{Enc}_{\mathcal{B}}(dT_i)$ and a NIZK proof V_k that
 - \mathcal{U}_k knows a signature C_k on u_k and w_k
 - $T_k = pk_{\mathcal{U}_k} g^{\frac{r_l}{u_k+dS+d_l}} = g_0^{u_k} g^{\frac{r_l}{u_k+dS+d_l}}$ and $r_k = g_0^{\frac{1}{u_k+d_k}}$,
 - d^2S, dT_j and d^2T_i are correct encryption of the unrevealed values dS, T_j and dT_i , respectively,
 - there exists a NIZK proof V_j proving that dS, T_j and dT_i are well-formed, using in particular r_k and d_k , and linked to valid signatures of the bank, the one from the withdrawal phase and the one corresponding to the certificate of \mathcal{U}_j (this step corresponds to a metaproof as described in [12]),
 without revealing $u_k, dS, C_j, w_k, T_j, dT_i$ nor V_j .
3. The spent coin is represented by $(d^2S, \pi = (T_k, dT_j, d^2T_i, V_k, r_l, d_l))$.

The next spendings of this coin work similarly and are not described in this paper.

Deposit and Identify. During a deposit, \mathcal{U} sends the received coin (e.g. of the form $(d^2S, \pi = (T_k, dT_j, d^2T_i, V_k, r_l, d))$) to \mathcal{B} . Then \mathcal{B} first checks if this coin is fresh by decrypting d^2S until obtaining the initial S and by testing if S already belongs to \mathcal{L} . If this is not the case, then everything is ok. Otherwise, there is a double-spending and \mathcal{B} has two deposited coins. Then, \mathcal{B} compares the first spending of both coins. If they are identical, then \mathcal{B} goes to the second one and so on until two spends at the same level are different (this case always happens). \mathcal{B} finally retrieves the identity of the cheater by first decrypting the related values T and T' and next using the compact e-cash technique to retrieve the cheater public key.

5.4 Achieving the PA₂* Property

Note that our PA₂* scheme is in accordance with the result of [6] which says that an unbounded adversary can always recognize his own coin during the game if it sees it later in a payment since such adversary is capable of decrypting values to retrieve the spender’s identity.

Theorem 3. *Under the security of the used encryption scheme (e.g. El Gamal), the security of NIZK proofs and the security of the Dodis-Yampolskiy PRF, the proposed scheme fulfils the PA₂* property.*

Sketch of Proof. Assume that $\mathcal{A}_{\text{PA}_2^*}$ succeeds in breaking the PA₂* property of the scheme described at Section 5.3. That means that $\mathcal{A}_{\text{PA}_2^*}$ is able to decide, between two honest users i_0 and i_1 chosen by $\mathcal{A}_{\text{PA}_2^*}$, which user is the spender i_b

during a call to the oracle $\text{Spd}(i_b)$. Note that, there is no restriction on the list of *authorized oracles* for such adversary.

The best strategy for $\mathcal{A}_{\text{PA}_2^*}$ is to choose the users i_0 and i_1 such that he has previously manipulated all the coins owned by these users. Then $\mathcal{A}_{\text{PA}_2^*}$ has to recognize the coin $C = (d^i S, \pi = (T_l, dT_k, \dots, d^i T_j, V_l, r_m, d_m))$ sent by i_b that he has previously owned. Consequently, $\mathcal{A}_{\text{PA}_2^*}$ knows some values that has been used to compute this coin (such as e.g. $d^{i_0} T_{j_0}$). When receiving a coin, $\mathcal{A}_{\text{PA}_2^*}$ cannot learn anything from:

- the encrypted serial number $d^i S$ under the security of the probabilistic encryption scheme (even if he knows the value that is encrypted),
- the encrypted security tags $dT_k, \dots, d^i T_j$ under the security of the encryption scheme (even if he knows some encrypted values),
- the security tag T_l of the spender under the security of the Dodis-Yampolskiy PRF (see [715] for details),
- the values r_m and d_m that comes from $\mathcal{A}_{\text{PA}_2^*}$ himself,
- the proof V_l by definition of a NIZK proof. In particular, see the result on metaproofs [12] and on usual zero-knowledge proofs of knowledge based on the discrete logarithms [84].

Consequently, even if $\mathcal{A}_{\text{PA}_2^*}$ has already seen the spent coin, he cannot recognize it. Thus, $\mathcal{A}_{\text{PA}_2^*}$ cannot win the PA_2^* game and our construction is PA_2^* , which concludes the proof. \square

We finally show that there is no relation between previously defined anonymity properties and the PA_2^* one.

Proposition 2. *PA_2^* and WA (resp. SA, resp. FA, resp. PA_1^*) are independent properties.*

Sketch of Proof. $\text{WA} \not\Rightarrow \text{PA}_2^*$. The scheme given in [1011] fulfils the WA property but not the PA_2^* property since the serial number of a coin is not protected and it does not change from one spend to another.

$\text{PA}_2^* \not\Rightarrow \text{WA}$. See appendix B.

$\text{PA}_2^* \not\Rightarrow \text{SA}$. This is due to $\text{SA} \Rightarrow \text{WA}$ and $\text{PA}_2^* \not\Rightarrow \text{WA}$.

$\text{SA} \not\Rightarrow \text{PA}_2^*$. The schemes proposed in [1465] fulfil the SA property but not the PA_2^* property since the serial number of a coin is not protected and it does not change from one spend to another.

$\text{PA}_2^* \not\Rightarrow \text{FA}$. This comes from $\text{FA} \Rightarrow \text{SA}$ and $\text{PA}_2^* \not\Rightarrow \text{SA}$.

$\text{FA} \not\Rightarrow \text{PA}_2^*$. The scheme proposed in Section 4.1 fulfils the FA property but not the PA_2^* property.

$\text{PA}_1^* \not\Rightarrow \text{PA}_2^*$. The generic construction given in Section 4.1 fulfils PA_1^* but not PA_2^* property (for obvious reasons).

$\text{PA}_2^* \not\Rightarrow \text{PA}_1^*$. The PA_2^* scheme proposed in Section 5.3 does not fulfil the PA_1^* property since the adversary being the bank in the PA_1^* game can decrypt all encrypted data of all spends. \square

6 Conclusion

In this paper, we provide the first study-in-depth of anonymity properties in transferable e-cash by introducing the full anonymity (FA) and perfect anonymity (PA). We show that the FA property can be reached by providing a generic construction and we prove that the PA cannot. We then define two restricted PA properties called PA_1^* and PA_2^* and we show that both restricted properties can be reached. Finally, we show that FA, PA_1^* and PA_2^* are three separate properties. Thus, an anonymous transferable e-cash scheme should ideally fulfil these three properties, which is the case (obviously inefficiently) if we apply the trick of Section 4.1 to the PA_2^* scheme of Section 5.3. Note that all our results can easily be extended to wallets by using the compact e-cash techniques 2.

Acknowledgments. We are grateful to Marc Girault, Pascal Paillier and Jacques Traoré for their suggestions of improvement, and to anonymous referees for their valuable comments.

References

1. Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In: STOC 1988, pp. 103–112. ACM Press, New York (1988)
2. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
3. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
4. Canard, S., Coisel, I., Traoré, J.: Complex Zero-Knowledge Proofs of Knowledge Are Easy to Use. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 122–137. Springer, Heidelberg (2007)
5. Canard, S., Gouget, A., Traoré, J.: Improvement of Efficiency in (Unconditional) Anonymous Transferable E-Cash. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 571–589. Springer, Heidelberg (2007)
6. Chaum, D., Pedersen, T.P.: Transferred Cash Grows in Size. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 390–407. Springer, Heidelberg (1993)
7. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
8. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, Springer, Heidelberg (2004)
9. Kim, Y., Perrig, A., Tsudik, G.: Communication-Efficient Group Key Agreement. In: IFIP/Sec 2001. IFIP Conference Proceedings, vol. 193, pp. 229–244. Kluwer, Dordrecht (2001)
10. Okamoto, T., Ohta, K.: Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 481–496. Springer, Heidelberg (1990)

11. Okamoto, T., Ohta, K.: Universal Electronic Cash. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 324–337. Springer, Heidelberg (1991)
12. De Santis, A., Yung, M.: Cryptographic Applications of the Non-Interactive Metaproof and Many-Prover Systems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 366–377. Springer, Heidelberg (1991)
13. Trolin, M.: A Stronger Definition for Anonymous Electronic Cash. In: ePrint Archive (2006)
14. van Antwerpen, H.: Electronic Cash. Master’s thesis, CWI (1990)

A Proof of Proposition 1: $FA \not\Rightarrow PA_1^*$

We first describe a toy scheme \mathcal{TS} and next we prove that \mathcal{TS} fulfils the FA property but it does not fulfil the PA_1^* property.

Description of the Toy Scheme \mathcal{TS} . We assume that \mathcal{S} is a transferable e-cash scheme that fulfils the SA property (e.g. [14,6,5]). We need to re-define only the spending protocol of \mathcal{S} . We additionally use as building blocks a secure symmetric encryption scheme $\mathcal{E} = (\text{Enc}, \text{Dec})$ and a unilateral authenticated group key agreement (GKA) scheme which uses e.g. the proposal of [9] where g is a public element. Each user has a signature key pair together with a valid certificate. In particular, this permits us to make the GKA scheme resistant to man-in-the-middle attacks. If \mathcal{U}_1 wants to spend a withdrawn coin to \mathcal{U}_2 , he has to proceed as follows.

- \mathcal{U}_1 chooses at random a value K_1 and sends g^{K_1} to \mathcal{U}_2 . User \mathcal{U}_2 chooses at random a value K_2 , computes g^{K_2} , signs $g^{K_1} \| g^{K_2}$ and sends g^{K_2} and the signature to \mathcal{U}_1 . Both can securely and secretly compute $K = g^{K_1 K_2}$ and execute a key-confirmation protocol.
- \mathcal{U}_1 and \mathcal{U}_2 play together the **Spend** protocol of \mathcal{S} by encrypting communications using the encryption algorithm **Enc** with the common secret key K . Both \mathcal{U}_1 and \mathcal{U}_2 can decrypt communications using the decryption algorithm **Dec** with the common secret key K .

If \mathcal{U}_2 wants to spend a received coin to \mathcal{U}_3 , the protocol is as follows.

- \mathcal{U}_2 sends $g^K = g^{g^{K_1 K_2}}$ to \mathcal{U}_3 . User \mathcal{U}_3 chooses at random a value K_3 , computes g^{K_3} , signs $g^K \| g^{K_3}$ and sends g^{K_3} and the signature to \mathcal{U}_2 . Both can compute $K' = g^{K K_3} = g^{g^{K_1 K_2} K_3}$, as in [9], and execute a key-confirmation protocol.
- \mathcal{U}_2 and \mathcal{U}_3 play together the **Spend** protocol of \mathcal{S} using **Enc** and the session key K' , as for the spending of a withdrawn coin

Note that the adversary playing the role of \mathcal{U}_2 cannot take any advantage in not sending the correct value $g^K = g^{g^{K_1 K_2}}$ for obvious reasons.

Proposition 3. *Under the assumptions that \mathcal{S} fulfils the SA property, and \mathcal{E} is secure, the \mathcal{TS} system fulfils the FA property.*

Sketch of Proof. Assume that \mathcal{A}_{FA} succeeds in breaking the FA property of \mathcal{TS} and thus decide, between i_0 and i_1 , which user is the user i_b from which \mathcal{A}_{FA} receives the coin of the challenge.

Note that the oracle Rcv is not allowed for the manipulation of users i_0 and i_1 . Thus, at Step 2 of the Game, \mathcal{A}_{FA} chooses users i_0 and i_1 such that all the coins owned by users i_0 and i_1 have never been owned by \mathcal{A}_{FA} . Then, \mathcal{A}_{FA} owns the coin of the challenge for the first time at step 4.

Before Step 3 of the game, \mathcal{A}_{FA} has observed the withdrawal of the coin of the challenge (using the oracle $\text{With}(U)$) and \mathcal{A}_{FA} may have observed many times a spending between two honest users involving the coin of the challenge, using the oracles $\text{Spd}\&\text{Rcv}$.

By assumption (\mathcal{S} fulfils the SA property), \mathcal{A}_{FA} cannot get the serial number of a coin involved in a withdrawal protocol. By construction of \mathcal{TS} , all communications related to the spending of a coin are encrypted with an anonymous ephemeral session key. Thus, all communications related to a call to the oracle $\text{Spd}\&\text{Rcv}$ are encrypted. That means that \mathcal{A}_{FA} has no information about the identifier of the coin embedded into the spending (\mathcal{A}_{FA} may know the entry number of the coin in \mathcal{OC} but not the serial number), except if \mathcal{A}_{FA} has succeeded in breaking either the security of the unilateral authenticated group key agreement or the security of \mathcal{E} to decrypt the communications without knowing the decryption key which is impossible by assumption. \square

Proposition 4. *\mathcal{TS} does not fulfil the PA_1^* property.*

Sketch of Proof. \mathcal{TS} does not fulfil the PA_1^* property (by construction). Indeed, $\mathcal{A}_{\text{PA}_1^*}$ can always choose one of his coins and give it to i_0 that has no coin. Since $\mathcal{A}_{\text{PA}_1^*}$ has received the coin, he necessarily knows the session key K . During the game, $\mathcal{A}_{\text{PA}_1^*}$ chooses i_0 as defined previously and i_1 at random. Then, the oracle $\text{Spd}\&\text{Rcv}(i_b, i)$, where i is a random honest user, is called. The underlying Spend protocol uses a session key K' from a key \tilde{K} introduced by i_0 or i_1 and a random key K_3 introduced by the receiver, as in the spending protocol. Then $\mathcal{A}_{\text{PA}_1^*}$ can easily check if the key \tilde{K} corresponds or not to the key K he knows. If this is the case, $\mathcal{A}_{\text{PA}_1^*}$ outputs 0 and he outputs 1 otherwise and wins the game with a probability of success equal to 1, which concludes the proof. \square

B Proof of Proposition 2: $\text{PA}_2^* \not\Rightarrow \text{WA}$

In order to prove that $\text{PA}_2^* \not\Rightarrow \text{WA}$, we describe a toy scheme and we prove that it fulfils the PA_2^* property but not the WA one.

Withdrawal protocol. The user \mathcal{U} gets from the bank \mathcal{B} a signature σ on the serial number s of the coin. Note that the serial number is not hidden to the bank that consequently knows s and σ .

Spending a withdrawn coin. The user \mathcal{U}_1 spends the coin (s, σ) to the user \mathcal{U}_2 by encrypting s and the signature σ to obtain E and producing a NIZK proof that the encrypted σ is a signature of the encrypted value s .

Spending a received coin. \mathcal{U}_2 spends a received coin (E, U) to \mathcal{U}_3 by using the metaproof technique [12] to produce a NIZK proof V that there exists a NIZK proof U of validity of the spent coin. This step can be done many times so that the coin can be spent again and again.

This scheme is straightforwardly PA_2^* but does not achieve the WA property since the bank can decrypt all spends to retrieve s and thus make the link with the initial withdrawal.

Generic Security-Amplifying Methods of Ordinary Digital Signatures

Jin Li¹, Kwangjo Kim¹, Fangguo Zhang², and Duncan S. Wong³

¹ International Research center for Information Security (IRIS)
Information and Communications University(ICU)
103-6 Munji-Dong, Yuseong-Gu, Daejeon, 305-732, Korea
{jjl,kkj}@icu.ac.kr

² School of Information Science and Technology
Sun Yat-Sen University, Guangzhou, 510275, P.R.China
isdzhfg@mail.sysu.edu.cn

³ Department of Computer Science
City University of Hong Kong, Hong Kong, China
duncan@cs.cityu.edu.hk

Abstract. We describe two new paradigms on how to obtain ordinary signatures that are secure against existential forgery under adaptively chosen message attacks (*fully-secure*, in short), from any signatures satisfy only a weak security notion called existentially unforgeable against weak chosen message attacks (*weakly-secure*, in short). The new transformations from a *weakly-secure* signature scheme to *fully-secure* signature scheme are generic, simple, and provably secure in the standard model. Moreover, these two new paradigms are built only on *weakly-secure* signatures. They are different from the previous methods, which also relied on some other cryptographic protocols or non-standard models.

By using two new paradigms, several efficient instantiations without random oracles are also presented, which are based on two previous *weakly-secure* signature schemes. These *fully-secure* signature schemes have many special interesting properties compared with the previous related signature schemes.

Keywords: Signature, Weak Chosen Message Attack, q -SDH Assumption, Strong-RSA Assumption, Strong Unforgeability.

1 Introduction

Digital signature is a central cryptographic primitive. The standard definition on the security of signature scheme was given by Goldwasser, Micali, and Rivest [18]. In fact, in terms of the goals and resources of the adversary, there are many security models can be formed. Compared to the standard security model, there are also many weak security models. Signatures in these weak security models are not sufficient in many practical applications. Among these weak security models, we will focus on the weak security model mentioned in [5,18], which is called existentially unforgeable against generic chosen message attack (or,

weak chosen message attack). The signatures that satisfy this security model are called *weakly-secure* signatures in this paper. In this security model, it requires the adversary to submit all signature queries before the signer's public key is published. More detailed definition will be given in Section 2. Obviously, because of the limitation of signature queries, it is insecure in many practical applications. However, in our paper, we will show its applications in the constructions of standard signatures and strongly secure signatures.

Since the standard definition on the security of signature schemes was given [18], there are many attempts to design practical and provably secure signature schemes in this security model. These methods can be divided into two categories, namely, concrete construction method and generic construction method.

There are many concrete constructions of signature schemes based on some standard assumptions, such as discrete logarithm problem [28,30], computational Diffie-Hellman problem [6,17,33], factoring [3]. Some constructions are based on other assumptions [29,34]. These schemes are very efficient, but their security can be proven only in the random oracle model. However, Canetti *et al.* [9] proved that some popular cryptosystems previously proved secure in the random oracle are actually provably insecure when the random oracle is instantiated by any real-world hashing functions. Over the years, several signature schemes were proposed in the standard model based on some stronger complexity assumptions such as [5,8,13,16]. Among them, the most efficient schemes are based on the Strong-RSA assumption [13,16] and q -strong Diffie-Hellman (q -SDH) assumption [5]. These assumptions are cryptographically stronger than the computational Diffie-Hellman, factoring, and RSA assumptions. The reason is that in order to simulate the signing oracle for the adversary in the proof, the simulator has to get some additional auxiliary inputs.

There are also many generic constructions of signatures. Most of them are based on the basic cryptographic primitive, such as (trapdoor) one-way functions [1,21]. There are also many generic constructions from other cryptographic protocols such as non-interactive zero-knowledge [19] and [12,15] *et al.* Among them, the most famous is the Fiat-Shamir (FS) transform [15]. Recall that the FS-transform [15] is a way to obtain a signature scheme from a three-move identification Σ protocol (honest-verifier zero-knowledge protocol) by collapsing the interaction via a hash function. But its security relies on the random oracle model. In order to avoid the usage of random oracle model, from the Σ protocol, Cramer *et al.* [12] gave another generic transform. The transform also uses hash function which does not act as a random oracle in the proving process. This conversion method is not practical because it used the authentication tree. Very recently, Bellare and Shoup [4] showed a simple transform for the construction of standard and strongly secure signatures from the Σ protocol, using the tool of two-tier signatures.

Compared to the standard security notion of existential unforgeability, there is another strong security notion which is called strongly existential unforgeability. Recently, this notion was concerned by many papers, such as [7,20,32].

1.1 Our Results

Firstly, we present two new paradigms to transform any *weakly-secure* signature schemes into *fully-secure* signature schemes. More precisely, the two paradigms are called sequential composition and parallel composition method, respectively. The new transformations from a *weakly-secure* signature scheme to *fully-secure* one are generic, simple and provably in the standard model. The goal is to obtain constructions that are based on standard assumptions and are efficient. They have interest from both theoretical and practical perspective.

- **Sequential Composition** (of weak signatures): This paradigm requires two weak signature schemes sequentially. Key pair in the first weak signature scheme is generated in key generation algorithm and used to sign the other public key, which is generated in signing algorithm.
- **Parallel Composition** (of weak signatures): Two weak signature schemes are also required in this paradigm, however, both of their key pairs should be generated in key generation algorithm, and used to sign two random and related messages.

We also show several efficient instantiations without random oracles converted from two *weakly-secure* signature schemes. The first paradigm, *i.e.*, the sequential composition method, is very efficient in key generation algorithm compared to the second. However, the signing algorithm is more efficient in the second paradigm. So, we can use different paradigm in different circumstances according to its requirements. This is a coincidence that, when instantiated from weak signature scheme [16], the construction will be similar to twin signature scheme [25]. In fact, our second paradigm can be viewed as generalization and extension of the twin signature scheme [25]. And, in both paradigms, if the signing algorithm in the weak signature is deterministic, the resulted *fully-secure* signature is strongly unforgeable secure.

1.2 Organization

In the next section, the definitions of variant signatures are given. Then, two previous instantiations of *weakly-secure* signature schemes are reviewed in Section 3. In Section 4, we propose our two generic transformations techniques. The security proof for these two transformations are given in Appendix. In Section 5, two instantiations from sequential composition method are presented based on two previous *weakly-secure* signature schemes. In Section 6, we present the two instantiations from parallel composition method. We discuss the efficiency of our two generic transformation methods and instantiations by comparing them with the previous signatures. Finally, the conclusion will be made.

2 Preliminaries

A signature scheme is defined by the following algorithms:

- *Key generation algorithm* Gen. On input 1^k , where k is the security parameter, it outputs (pk, sk) as public and secret keys.

- *Signing algorithm* **Sign**. On input a message m and sk , it outputs a signature σ .
- *Verification algorithm* **Verify**. Given public key pk , message m and signature σ , algorithm **Verify**(pk, m, σ) outputs 1 if $\sigma \leftarrow \text{Sign}(sk, m)$. Otherwise, output 0.

In terms of the goals of the adversary, it can be divided into four categories [18]:

- *Total break*: This is the most serious attack, in which the adversary is able to disclose the secret key of the signer.
- *Universal forgery*: The adversary is able to sign any given messages.
- *Existential forgery*: The adversary is able to provide a signature on a new message whose signature has not been seen.
- *Strong Existential forgery*: The adversary is able to provide a new message-signature pair.

On the other hand, various resource can be made available to the adversary, helping into his/her forgery [18]. We focus ourselves on two kinds of message attacks:

- *Weakly chosen message attack*: The adversary is allowed to obtain signatures from the signer for a chosen list of messages before it attempts to break the scheme. These messages chosen by the adversary must be given to the signer before seeing the signer's public key.
- *Adaptively chosen message attack*: The adversary is allowed to request signatures of messages chosen by itself. These messages may not only depend on signer's public key, but also depend on the previous obtained signatures.

2.1 Unforgeability

By combining the different goals of the adversary and various resource available to the adversary, many security notions for signature schemes can be derived. The standard notion of security for a signature scheme is called existential unforgeability under adaptively chosen message attacks (*fully-secure* signatures) [18], which is defined through the following game between a challenger \mathcal{C} and an adversary \mathcal{A} :

Setup: A public/private key pair $(pk, sk) \leftarrow \text{Gen}(1^k)$ is generated and adversary \mathcal{A} is given the public key pk .

Query: \mathcal{A} runs for time t and issues q signing queries to a signing oracle in an adaptive manner, that is, for each i , $1 \leq i \leq q$, \mathcal{A} chooses a message m_i based on the message-signature pairs that \mathcal{A} has already seen, and obtains in return a signature σ_i on m_i from the signing oracle (*i.e.*, $\sigma_i = \text{Sign}(sk, m_i)$).

Forge: \mathcal{A} outputs a forgery (m^*, σ^*) and halts. \mathcal{A} wins if

- σ^* is a valid signature on message m^* under the public key pk , *i.e.*, $\text{Verify}(pk, m^*, \sigma^*) = 1$; and
- m^* has never been queried, *i.e.*, $m^* \notin \{m_1, m_2, \dots, m_q\}$.

Definition 1 (Unforgeability). A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ is (t, q, ε) -fully-secure, if any adversary with run-time t wins the above game with probability at most ε after issuing at most q signing queries.

If we lower down the adversary's goal to strong existential forgeability and keep its ability unchanged, we can get a stronger definition compared to existential unforgeability against adaptive chosen message attacks:

2.2 Strong Existential Unforgeability

The notion is also defined using the above game between a challenger \mathcal{C} and an adversary \mathcal{A} , except the definition that “ \mathcal{A} wins the game” is \mathcal{A} can output a pair (m^*, σ^*) such that (m^*, σ^*) does not belong to the previous queried set $\{(m_i, \sigma_i)\}$ and $\text{Verify}(pk, m^*, \sigma^*) = 1$.

If we lower the adversary's ability to weak chosen message attack while keeping the goal of the adversary unchanged compared to the standard security notion, we can get a weaker definition compared to existential unforgeability against adaptive chosen message attacks:

2.3 Weak Unforgeability

The difference between this security notion with the standard security [18] is that here it requires that the adversary should submit all messages for signature queries before the public key is seen. And we define “ \mathcal{A} wins the game” is equivalent to \mathcal{A} can output a pair (m^*, σ^*) such that σ is a valid signature of a new message m^* .

Pre-Proceeding: Adversary \mathcal{A} runs for time t and issues q signing queries to a signing oracle, i.e., \mathcal{A} chooses messages m_i , where $1 \leq i \leq q$.

Setup: A public/private key pair $(pk, sk) \leftarrow \text{Gen}(1^k)$ is generated and adversary \mathcal{A} is given the public key pk . Meanwhile, q signatures σ_i on m_i from the signing oracle (i.e., $\sigma_i = \text{Sign}(sk, m_i)$), are also returned to \mathcal{A} .

Forge: \mathcal{A} outputs a forgery (m^*, σ^*) and halts. \mathcal{A} wins if

- σ^* is a valid signature on message m^* under the public key pk , i.e., $\text{Verify}(pk, m^*, \sigma^*) = 1$; and
- m^* has never been queried, i.e., $m^* \notin \{m_1, m_2, \dots, m_q\}$.

Definition 2 (Weak Unforgeability). A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ is (t, q, ε) -weakly-secure, if any adversary with run-time t wins the above game with probability at most ε .

3 Instantiations of Weak Signatures

It has been shown in [5,16] that two *weakly-secure* signature schemes can be constructed, based on the q -SDH assumption and Strong-RSA assumption, respectively, in the standard model.

3.1 Weak Boneh-Boyen Signature [5]

Before describing the weak Boneh-Boyen signature, we first introduce some preliminaries on bilinear maps and an assumption used in [5].

Let \mathbb{G}_1 and \mathbb{G}_2 be cyclic groups of prime order p with the multiplicative group action. And, g is a generator of \mathbb{G}_1 . Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a map with the following properties:

1. Bilinearity: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for all $g_1, g_2 \in \mathbb{G}_1$, and $a, b \in_{\mathbb{R}} \mathbb{Z}_p$;
2. Non-degeneracy: There exists $g_1, g_2 \in \mathbb{G}_1$ such that $\hat{e}(g_1, g_2) \neq 1$, in other words, the map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in \mathbb{G}_2 ;
3. Computability: There is an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}_1$.

As shown in [6,34], such non-degenerate bilinear maps over cyclic groups can be obtained from the Weil or the Tate pairing over algebraic curves.

Definition 3. (*q-Strong Diffie-Hellman Assumption (q-SDH in short)*). *The q-SDH assumption in group \mathbb{G}_1 is defined as follows: given a $(q + 1)$ -tuple $(g, g^x, g^{x^2}, \dots, g^{x^q}) \in (\mathbb{G}_1)^{q+1}$ as input, it is hard to output a pair $(c, g^{1/(x+c)})$, where $c \in \mathbb{Z}_p^*$.*

Next, we describe the weak Boneh-Boyen signature [5]. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear groups where the order of \mathbb{G}_1 and \mathbb{G}_2 is p . As usual, g is a generator of \mathbb{G}_1 .

1. **Gen:** Pick $x \in \mathbb{Z}_p^*$, compute $y = g^x$. The public key is y and the secret key is x .
2. **Sign:** Given message $m \in \mathbb{Z}_p^*$, the signer outputs the signature on m as $\sigma = g^{\frac{1}{x+m}}$.
3. **Verify:** On input verification key y , message m , and the signature σ , output 1 if and only if $\hat{e}(y \cdot g^m, \sigma) = \hat{e}(g, g)$. Otherwise, output 0.

Theorem 1. *The weak Boneh-Boyen signature is weakly-secure if the q-SDH assumption holds.*

Proof. Refer to [5]. □

3.2 Weak GHR Signature [16]

Gennaro, Halevi and Rabin proposed a secure signature scheme [16] (denoted by GHR signature) without random oracle, however, under the assumption that hash function H is division intractable, and acts like the random oracle model or achieves the chameleon property, which was called a non-standard randomness-finding oracle in [16]. Division intractability means that it is computationally impossible to find a_1, a_2, \dots, a_k and b such that $H(b)$ divides the product of all the $H(a_i)$. In order to get a *fully-secure* signature without random oracles, the non-standard randomness-finding oracle was required [16]. This non-standard assumption helps the simulator to find the second preimage during the simulation. The randomness-finding oracle is non-standard because it requires that,

given a hash function H , values M and e , one could find a random value R such that $H(R, M) = e$. In fact, without the assumption of randomness-finding oracle, the simulator has to guess which messages the adversary will ask during the signing simulation phase. Then, the scheme in [16] can only be proven *weakly-secure* without the randomness-finding oracle. This problem was also addressed in [11], which presented an extension to [16] without relying on this non-standard assumption.

Definition 4. (Strong-RSA Assumption) *Given a randomly chosen RSA modulus n , and a random element $s \in \mathbb{Z}_n^*$, it is infeasible to find a pair (e, r) with $e > 1$ such that $r^e = s \pmod{n}$.*

We describe the weak GHR signature scheme as follows:

1. **Gen:** Pick two safe primes p and q , compute $n = pq$ as RSA modulus, a hash function H , and select $s \in \mathbb{Z}_n^*$. The public key is (n, s) and the secret key is (p, q) .
2. **Sign:** To sign a message m , the signer computes $e \leftarrow H(m)$ and outputs the signature as $\sigma = s^{\frac{1}{e}} \pmod{n}$.
3. **Verify:** On input verification key (n, s) , message m , and σ , output 1 if and only if $\sigma^{H(m)} = s \pmod{n}$. Otherwise, output 0.

Theorem 2. *The weak GHR signature scheme is weakly-secure if the Strong-RSA assumption holds and H is division intractability.*

Proof. Refer to [10,16] □

As stated in [11,16], division-intractable hash functions can be constructed from collision-intractable hash functions [26].

4 Fully-Secure Signatures from Weakly-Secure Signatures

4.1 Related Work

There are two main techniques in order to get *fully-secure* signatures from *weakly-secure* signatures:

- Random Oracle Model: By using the hash function on the messages for signatures without changing other algorithms, the new signatures can be fully-secure from the back patch property of random oracle [2]. This method was used in [5,34].
- Chameleon Hash Function: By combining weakly-secure signatures with the chameleon hash function, the signer can first sign any value with the weak signature scheme. Then it can sign the real message from the signature on any value, by using the property of chameleon hash function. Many papers have used this technique, such as [5,14,24,31].

Compared to *fully-secure* signatures, the construction of *weakly-secure* signatures is relatively easy (Obvious, every *fully-secure* signature scheme also satisfies security notion of *weakly-secure* signature). There have several *weakly-secure* signature schemes in the open literature.

4.2 Sequential Composition Method

Given a *weakly-secure* signature scheme $\Pi' = (\text{Gen}', \text{Sign}', \text{Verify}')$, we construct a *fully-secure* signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ by using the sequential composition method. We assume that the public key space belongs to the message space in this paradigm. Otherwise, hash function or other techniques could be applied here to achieve this. The construction of Π proceeds as follows:

- **Gen.** On input security parameter 1^k , invoke $\text{Gen}'(1^k)$ and obtain $(pk, sk) \leftarrow \text{Gen}'(1^k)$. Output Π 's public key pk and secret key sk (In fact, $\text{Gen} = \text{Gen}'$).
- **Sign.** To sign message m , the signer first invokes $\text{Gen}'(1^k)$ to obtain a key pair $(pk', sk') \leftarrow \text{Gen}'(1^k)$. The signer then invokes algorithms $\text{Sign}'(sk, pk')$ and $\text{Sign}'(sk', m)$. Finally, it outputs $\sigma = (A, B, C)$ as the signature, where $A = \text{Sign}'(sk, pk')$, $B = \text{Sign}'(sk', m)$, $C = pk'$.
- **Verify.** On input verifying key pk , message m , and signature $\sigma = (A, B, C)$, output 1 if and only if $\text{Verify}'(pk, C, A) = 1$ and $\text{Verify}'(C, m, B) = 1$.

Key generation of the resulted *fully-secure* signature Π is the same with the key generation of weak signature Π' . In signature generation phase, $\text{Sign}'(sk, pk')$ can be pre-computed by the signer. The construction is similar with [4,22]. However, only *weakly-secure* signatures are required here, instead of fully secure signature scheme or one-time signature scheme as required in [4,22]. This could be viewed as improvements to the results [4,22].

Below, we formally prove the security of the signature scheme Π . We denote the cost of a signing algorithm Sign' in Π' by $t_{\text{sign}'}$.

Theorem 3. *If Π' is (t', q, ϵ') -weakly-secure, then the signature Π is (t, q, ϵ) -fully-secure, where $t \leq t' - O(q \cdot t_{\text{sign}'})$ and $\epsilon \geq 2q \cdot \epsilon'$. □*

Proof. See Appendix A. □

In fact, if the signing algorithm Sign' in Π' deterministic, then the *fully-secure* signature scheme Π is strongly unforgeable.

4.3 Parallel Composition Method

In this section, we show another generic transformation from *weakly-secure* signatures to *fully-secure* signatures.

Before showing the transformation, we define a relation $\mathcal{R} = \{((a, b), c)\}$ that satisfies the following conditions:

- Given a and c (or b and c), b (or a) is determined and can be computed in probabilistic polynomial time (*PPT*);

¹ In fact, the key pair $(pk', sk') \leftarrow \Pi'$ generated in the signing algorithm is only used to sign message for one-time. So, in fact, (pk', sk') can be generated from weakly-secure signatures satisfies only $(t', 1, \epsilon')$ -weakly-secure, which is easier to be constructed compared to (t', q, ϵ') -weakly-secure signatures.

- Given randomly chosen values a and b , it is hard to find c in PPT , such that $((a, b), c) \in \mathcal{R}$.

In fact, this kind of relation can be easily found. Suppose the security parameter is 1^k . For example, given a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$, $a, b \in \{0, 1\}^k$ and $c \in \{0, 1\}^*$, we define $((a, b), c) \in \mathcal{R}$, if and only if $a \oplus b = H(c)$.

Obviously, this relation satisfies the definition of \mathcal{R} because: Given $a \in \{0, 1\}^k$ and $c, b \in \{0, 1\}^k$ is determined and can be computed efficiently; And, randomly choose $a \in \{0, 1\}^k$ and $b \in \{0, 1\}^k$, it is hard to find c such that $a \oplus b = H(c)$ for the collision-resistant property of the hash function.

In public parameters, relation $\mathcal{R}=\{((a, b), c)\}$ defined above should be given. The generic construction follows:

1. **Gen.** On input security parameter 1^k , invoke $\text{Gen}'(1^k)$ two times and obtain two key pairs (pk_1, sk_1) and (pk_2, sk_2) . Output Π 's public key $pk = (pk_1, pk_2)$ and secret key $sk = (sk_1, sk_2)$.
2. **Sign.** To sign message m , the signer first chooses m' randomly and computes m'' such that $((m', m''), m) \in \mathcal{R}$. The signer then invokes algorithms $\text{Sign}'(sk_1, m')$ and $\text{Sign}'(sk_2, m'')$. Output $\sigma=(A, B, C)$ as the signature on message m , where $A = \text{Sign}'(sk_1, m')$, $B = \text{Sign}'(sk_2, m'')$, $C = m'$.
3. **Verify.** On input verifying key $pk = (pk_1, pk_2)$, message m , and signature $\sigma = (A, B, C)$, first compute m'' from m and C such that $((C, m''), m) \in \mathcal{R}$ (This can be done from the property of the relation R). Finally, it outputs 1 if and only if $\text{Verify}'(pk_1, C, A) = 1$ and $\text{Verify}'(pk_2, m'', B) = 1$.

It is easy to prove that Π is strongly unforgeable if Π' is deterministic. Below, we formally prove the security of the resulting signature scheme Π , with very tight security reduction to Π' . We also denote the cost of a signing algorithm sign' in Π' by $t_{\text{sign}'}$.

Theorem 4. *The signature scheme Π is (t, q, ε) -fully-secure, provided that Π' is (t', q, ε') -weakly-secure, where $t \leq t' - O(q \cdot t_{\text{sign}'})$ and $\varepsilon \geq 2\varepsilon'$.*

Proof. See Appendix B. □

4.4 Comparison of Two Paradigms

- Key generation phase: The key generation in *fully-secure* signature from the sequential method, is the same with its corresponding key generation of weak signature scheme. And, for the fully secure signature from parallel method, it requires to run the key generation algorithm of weak signature twice. So, the key size is smaller and computation cost is less in sequential method, compared with the parallel method.
- Signing phase: In the first paradigm, the signer should run the key generation algorithm and signing algorithm of weak signature, respectively. In the

second paradigm, it requires to run the signing algorithm of weak signature twice. The online computations of both methods in signing phase requires to run signing algorithm of weak signature only once.

- Verification phase: In both paradigms, it requires to run the verification of weak signature scheme twice. So, the computations of verification algorithm are the same.

In conclusion, the sequential method is more suitable for device with small storage such as smart card for its smaller key size. And, the signing algorithm in the sequential composition method requires one key generation of weak signatures. So, if the computation of this phase is almost the same with signing algorithm of weak signature, then, the sequential method is indeed better than the parallel composition method. Otherwise, from only the computational cost of signing algorithm, the parallel composition method is better. So, we can use different paradigms according to circumstance requirements.

After presenting two paradigms, we will describe several instantiations converted from the *weakly-secure signature* schemes [5,16].

5 Instantiations from Sequential Composition Method

5.1 Fully-Secure Signature from Weak Boneh-Boyen Signature

We describe how to get *fully-secure* signature, denoted by S-WBB, by using the sequential composition method on the weak Boneh-Boyen signature scheme. The public parameters are similar with the weak Boneh-Boyen signature, except a collision resistant hash function $H : \mathbb{G}_1 \rightarrow \mathbb{Z}_p^*$ is chosen additionally.

1. **Gen:** Pick $x \in \mathbb{Z}_p^*$, compute $y = g^x$. The public key is y and the secret key is x .
2. **Sign:** Given message $m \in \mathbb{Z}_p^*$, the signer chooses a random $x' \in \mathbb{Z}_p^*$, computes $y' = g^{x'}$, and outputs the signature as $\sigma=(A, B, C)$, where $A = g^{\frac{1}{x+H(y')}}$, $B = g^{\frac{1}{x'+m}}$, $C = y'$.
3. **Verify:** On input verification key y , message m , and the signature $\sigma = (A, B, C)$, output 1 if and only if $\hat{e}(y \cdot g^{H(C)}, A) = \hat{e}(g, g)$ and $\hat{e}(y' \cdot g^m, B) = \hat{e}(g, g)$. Otherwise, output 0.

In key generation algorithm, S-WBB scheme needs one exponentiation in group \mathbb{G}_1 . The signing algorithm costs two exponentiations computations in group \mathbb{G}_1 and two inversion computations in \mathbb{Z}_p^* . As the value A could be pre-computed, the computations is reduced to only one exponentiation in \mathbb{G}_1 and one inversion computation in \mathbb{Z}_p^* . In verification algorithm, the value $\hat{e}(g, g)$ can be fixed and published as part of the public key. So, it only needs two pairing and two exponentiations computations.

Compared with the *fully-secure* signature scheme in [5], the key generation algorithm of S-WBB is more efficient. Furthermore, the key size is smaller than [5] because the secret key consists of only one group element. So, it is very suitable

for small storage device such as smart card or mobile phone to perform authentication operations. The online computation for signing algorithm in [5] is also one exponentiation in \mathbb{G}_1 and one inversion computation in \mathbb{Z}_p^* . The computation of online verification in S-WBB requires one more pairing computation compared with [5]. From the above comparison, the S-WBB scheme is very suitable for device with small storage.

Theorem 5. *The S-WBB signature scheme is fully-secure.*

Proof. The result can be derived directly from Theorems 1 and 3. □

5.2 Fully-Secure Signature from Weak GHR Signature

In this section, we present a *fully-secure* signature, denoted by S-WGHR, from the weak GHR signature scheme [16].

1. **Gen:** On input security parameter 1^k , pick two pairs safe primes (p_1, q_1) . Compute $n_1 = p_1q_1$ as a RSA modulus, select $s_1 \in \mathbb{Z}_{n_1}^*$. Meanwhile, choose a division intractability hash function $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_{n_1}^*$. The public key is $(n_1, s_1, n_2, s_2, H_1)$ and the secret key is (p_1, q_1) .
2. **Sign:** To sign a message m , choose two pairs safe primes (p_2, q_2) , and a random $s_2 \in \mathbb{Z}_{n_2}^*$, compute $n_2 = p_2q_2$. Then, choose a division intractability hash functions and $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_{n_2}^*$ and compute the signature as $\sigma=(A, B, C)$, where $A = s_1^{\frac{1}{H_1(n_2 \| s_2 \| H_2)}} \bmod n_1$, $B = s_2^{\frac{1}{H_2(m)}} \bmod n_2$, $C = n_2 \| s_2 \| H_2$.
3. **Verify:** On input verification key (n_1, s_1, H_1) , message m , and $\sigma=(A, B, C)$, parse $C = (C_1, C_2, C_3)$. Then, output 1 if and only if $A^{H_1(C)} = s_1 \bmod n_1$ and $B^{C_3(m)} = C_2 \bmod C_1$. Otherwise, output 0.

Theorem 6. *The S-WGHR signature scheme is fully-secure.*

Proof. The result can be derived directly from Theorems 2 and 3. □

In key generation algorithm, it requires one multiplications in $\mathbb{Z}_{n_1}^*$. The secret key size is only $\lceil \log_2 n_1 \rceil$. The signing algorithm needs one exponentiation and inversion computations in $\mathbb{Z}_{n_1}^*$ and $\mathbb{Z}_{n_2}^*$, respectively. As the value A could be pre-computed, the computation is reduced to only one exponentiation and one inversion computation in $\mathbb{Z}_{n_2}^*$. In verification algorithm, it requires one exponentiation computation in $\mathbb{Z}_{n_1}^*$ and $\mathbb{Z}_{n_2}^*$, respectively. Compared to [25], the computations in signing and verification algorithms are almost the same. In key generation algorithm of S-WGHR, the key size is smaller than [25] and it requires less exponentiations to generate key pair.

6 Instantiations from Parallel Composition Method

In the following two instantiations, we will use the concrete relation \mathcal{R} given in Section 4.3: $((a, b), c) \in \mathcal{R}$, if and only if $a \oplus b = H(c)$. The relation should be described in system public parameters, in both following examples.

6.1 Fully-Secure Signature from Weak Boneh-Boyen Signature

Denote the following fully-secure signature scheme from the weak Boneh-Boyen by P-WBB. The public parameters are similar with the weak Boneh-Boyen signature, excluding a concrete relation \mathcal{R} given in Section 4.3.

1. **Gen:** Pick $x_1, x_2 \in \mathbb{Z}_p^*$, compute $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$. The public key is (y_1, y_2) and the secret key is (x_1, x_2) .
2. **Sign:** Given message $m \in \mathbb{Z}_p^*$, the signer chooses a random $m' \in \mathbb{Z}_p^*$ and computes the signature as $\sigma=(A, B, C)$, where $A = g^{\frac{1}{x_1+m'}}$, $B = g^{\frac{1}{x_2+(H(m)\oplus m')}}$, $C = m'$.
3. **Verify:** On input verification key (y_1, y_2) , message m , and the signature $\sigma = (A, B, C)$, output 1 if and only if $\hat{e}(y_1 \cdot g^C, A) = \hat{e}(g, g)$ and $\hat{e}(y_2 \cdot g^{H(m)\oplus C}, B) = \hat{e}(g, g)$. Otherwise, output 0.

In key generation algorithm of P-WBB, it needs two exponentiations in group \mathbb{G}_1 . The signing algorithm costs two exponentiations computations in group \mathbb{G}_1 and two inversion computations in \mathbb{Z}_p^* . In verification algorithm, it only needs two pairing and two exponentiations computations as the value $\hat{e}(g, g)$ can be published as part of the public key..

From Theorems 1 and 4, we can get the following result:

Theorem 7. *The P-WBB signature scheme is fully-secure.*

The security reduction is the same with Theorem 4.

6.2 Fully-Secure Signature from Weak GHR Signature

In this section, we present a *fully-secure* signature, denoted by P-WGHR, from the weak GHR signature[16] with the following advantages: The new scheme does not require the non-standard randomness-finding oracle assumption [16]. The signing algorithm requires less exponentials computation compared to [16].

1. **Gen:** On input security parameter 1^k , pick two pairs safe primes (p_1, q_1) , (p_2, q_2) . Compute $n_1 = p_1q_1$ and $n_2 = p_2q_2$ as two RSA modulus, select $s_1 \in \mathbb{Z}_{n_1}^*$ and $s_2 \in \mathbb{Z}_{n_2}^*$. Meanwhile, choose two division intractability hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_{n_1}^*$ and $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_{n_2}^*$. Furthermore, a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ is selected. The public key is $(n_1, s_1, n_2, s_2, H_1, H_2, H)$ and the secret key is (p_1, q_1, p_2, q_2) .
2. **Sign:** To sign a message m , the signer chooses a random $m' \in \{0, 1\}^k$ and computes the signature as $\sigma=(A, B, C)$, where $A = s_1^{\frac{1}{H_1(m')}} \bmod n_1$, $B = s_2^{\frac{1}{H_2(H(m)\oplus m')}} \bmod n_2$, $C = m'$.
3. **Verify:** On input verification key $(n_1, s_1, n_2, s_2, H_1, H_2, H)$, message m , and $\sigma=(A, B, C)$, output 1 if and only if $A^{H_1(C)} = s_1 \bmod n_1$ and $B^{H_2(H(m)\oplus C)} = s_2 \bmod n_2$. Otherwise, output 0.

It requires one multiplication in $\mathbb{Z}_{n_1}^*$ and $\mathbb{Z}_{n_2}^*$ in key generation algorithm, respectively. The signing algorithm needs one exponentiation and inversion computations in $\mathbb{Z}_{n_1}^*$ and $\mathbb{Z}_{n_2}^*$, respectively. The online computation in signing phase could be reduced to only one exponentiation and one inversion computation in $\mathbb{Z}_{n_2}^*$. In verification algorithm, it requires one exponentiation computation in $\mathbb{Z}_{n_1}^*$ and $\mathbb{Z}_{n_2}^*$, respectively.

It is very interesting because this instantiation from the weak GHR signature scheme looks similar to the twin signature scheme in [25]. In fact, the parallel composition paradigm could be viewed as generalization of [25]. First, we define a relation R as follows:

$(a, b), c) \in \mathcal{R}$ if and only if $a = c \oplus \mu_1 \parallel c \oplus \mu_2$, $b = \mu_1 \parallel \mu_2$ for some μ_1 and μ_2 .

It is easy to verify such kind of relation satisfies the definition given in Section 4.3. Based on this given relation and the parallel paradigm, the twin signature scheme [25] could be derived directly from the weak GHR signature scheme.

And, the following result could be derived easily from Theorems 2 and 4. And, security reduction is the same with Theorem 4.

Theorem 8. *The P-WGHR signature scheme is fully-secure.*

7 Conclusion

We showed two new paradigms on how to obtain *fully-secure* signature scheme from any scheme satisfies only a weak security notion called existentially unforgeable against generic chosen message attacks in the standard model. The new paradigms are different from known methods because they are built only on *weakly-secure* signatures, and do not rely on other cryptographic protocols such as one-time signature, or non-standard assumptions such as random oracle model. The transformations are simple, generic, and provably secure in the standard model. The sequential composition method is very efficient in key generation algorithm compared to the second. However, if the computation cost in the key generation algorithm of weak signature needs more than the weak signature's signing algorithm, then, the signing algorithm is more efficient in the second paradigm. So, we can use different paradigm in applications according to different requirements.

We also presented several concrete *fully-secure* signature schemes without random oracles converted from two previous *weakly-secure* signature schemes. Their efficiency comparison with the previous secure signatures was also given.

The design of existentially unforgeable secure signature scheme under adaptively chosen message attack, then, can be reduced to that of signature scheme which is secure under only weakly chosen message attack. In the standard model, constructing an efficient signature scheme based on standard assumption is still an open problem. The two new paradigms give one direction in order to solve this problem.

Acknowledgements

This work was partially supported by the 2nd stage of Brain Korea 21 Project sponsored by the Ministry of Education and Human Resources Development, Korea. The third author was supported by the National Natural Science Foundation of China (No. 60773202 and 60633030) and 973 Program (2006CB303104).

References

1. Bellare, M., Micali, S.: How to Sign Given Any Trapdoor Function. *J. of the ACM* 39, 214–233 (1992)
2. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: *ACM CCS 1993*, pp. 62–73. ACM Press, New York (1993)
3. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures-How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) *EUROCRYPT 1996*. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
4. Bellare, M., Shoup, S.: Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir without Random Oracles. In: Okamoto, T., Wang, X. (eds.) *PKC 2007*. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)
5. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
6. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from The Weil Pairing. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
7. Boneh, D., Shen, E., Waters, B.: Strongly Unforgeable Signatures Based on Computational Diffie-Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) *PKC 2006*. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
8. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
9. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. In: *STOC 1998*, pp. 207–221. ACM, New York (1998)
10. Chevallier-Mames, B., Joye, M.: A Practical and Tightly Secure Signature Scheme without Hash Function. In: Abe, M. (ed.) *CT-RSA 2007*. LNCS, vol. 4377, pp. 339–356. Springer, Heidelberg (2006)
11. Coron, J.-S., Naccache, D.: Security Analysis of The Gennaro-Halevi-Rabin Signature Scheme. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 91–101. Springer, Heidelberg (2000)
12. Cramer, R., Damgård, I.: Secure Signature Schemes Based on Interactive Protocols. In: Coppersmith, D. (ed.) *CRYPTO 1995*. LNCS, vol. 963, pp. 297–310. Springer, Heidelberg (1995)
13. Cramer, R., Shoup, V.: Signature Schemes Based on the Strong RSA Assumption. *ACM TISSEC* 3(3), 161–185 (2000); Extended abstract. In: *Sixth ACM Conference on Computer and Communication Security* (1999)
14. Even, S., Goldreich, O., Micali, S.: On-Line/Off-Line Digital Signatures. *Journal of Cryptology* 9, 35–67 (1996)
15. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) *CRYPTO 1986*. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)

16. Gennaro, R., Halevi, S., Rabin, T.: Secure Hash-and-Sign Signatures without The Random Oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
17. Goh, E.-J., Jarecki, S.: A Signature Scheme as Secure as The Diffie-Hellman Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 401–415. Springer, Heidelberg (2003)
18. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Computing* 17(2), 281–308 (1988)
19. Goldwasser, S., Ostrovsky, R.: Invariant Signatures and Non-Interactive Zero-Knowledge Proofs Are Equivalent. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 228–239. Springer, Heidelberg (1993)
20. Huang, Q., Wong, D.S., Zhao, Y.: Generic Transformation to Strongly Unforgeable Signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007)
21. Lamport, L.: Constructing Digital Signatures from a One Way Function. Technical Report CSL-98, SRI International (1979)
22. Li, J., Chan, Y.Y., Wang, Y.: A Generic Construction of Secure Signatures Without Random Oracles. In: GavriloVA, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3982, pp. 309–317. Springer, Heidelberg (2006)
23. Lindell, Y.: A Simpler Construction of CCA2-Secure Public Key Encryption under General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 241–254. Springer, Heidelberg (2003)
24. Krawczyk, H., Rabin, T.: Chameleon Hashing and Signatures. In: Proc. of NDSS 2000, Internet Society (1998), <http://eprint.iacr.org/1998/010>
25. Naccache, D., Pointcheval, D., Stern, J.: Twin Signatures: An Alternative to The Hash-and-Sign Paradigm. In: ACM Conference on Computer and Communications Security 2001, pp. 20–27. ACM, New York (2001)
26. Naor, M., Yung, M.: Universal One-Way Hash Functions and Their Cryptographic Applications. In: ACM symposium on Theory of Computing, pp. 33–43. ACM Press, New York (1989)
27. Perrig, A.: The BiBa One-Time Signature and Broadcast Authentication Protocol. In: Eighth ACM Conference on Computer and Communication Security, pp. 28–37. ACM, New York (2001)
28. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology* 13(3), 361–396 (2000)
29. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signature and Public Key Cryptosystems. *Comm. of ACM*, 120–126 (1978)
30. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. *Journal of Cryptology* 4, 161–174 (1991)
31. Shamir, A., Tauman, Y.: Improved Online/Offline Signature Schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
32. Steinfeld, R., Pieprzyk, J., Wang, H.: How to Strengthen Any Weakly Unforgeable Signature into a Strongly Unforgeable Signature. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 357–371. Springer, Heidelberg (2006)
33. Waters, B.: Efficient Identity-Based Encryption without Random Oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
34. Zhang, F., Safavi-Naini, R., Susilo, W.: An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)

Appendix A: Proof of Theorem 3

Proof. Given any adversary \mathcal{A} attacking Π in an adaptive chosen message attack, we construct an adversary \mathcal{A}' breaking Π' in weak chosen message attacks. After given public key pk of Π , \mathcal{A} queries the signing oracle of Π on messages m_i adaptively and gets q signatures $\sigma_i=(A_i, B_i, C_i)$ for $1 \leq i \leq q$. After the signature queries, \mathcal{A} outputs a forged signature on a new m^* as $\sigma^* = (A^*, B^*, C^*)$.

There are two types of forgeries.

Type 1 forgery: $C^* \neq C_i$ for $1 \leq i \leq q$.

Type 2 forgery: $C^* = C_i$ for some $i, 1 \leq i \leq q$.

The reduction works differently for each forger type. Therefore, initially \mathcal{A}' will choose a random bit $b_{code} \in \{1, 2\}$ that indicates its guess for the type of forger. The simulation proceeds differently for each b_{code} .

If $b_{code} = 1$, we construct an algorithm \mathcal{A}' to break Π' as follows:

Simulation of Key Generation. \mathcal{A}' first invokes $\text{Gen}'(1^k)$ and gets q key pairs $(pk_i, sk_i) \leftarrow \text{Gen}'(1^k)$ (Assume that \mathcal{A} makes at most q queries to signing oracle), and sends the q values pk_i , for $1 \leq i \leq q$, to challenger for signature queries of Π' before the parameters publication of Π' . Then \mathcal{A}' gets public key pk of Π' and q signatures $\sigma'_i = \text{Sign}'(sk, pk_i)$ on the q messages pk_i , with respect to pk , for $1 \leq i \leq q$. \mathcal{A}' sends the public key pk to the adversary \mathcal{A} as the public key of Π .

Simulation of Signing Oracle. \mathcal{A} then queries the signing oracle of Π on messages m_i adaptively for $1 \leq i \leq q$. \mathcal{A}' answers the signature query as $\sigma_i=(A_i, B_i, C_i)$, where $A_i=\sigma'_i$ from the challenger, $B_i = \text{Sign}'(sk_i, m_i)$, $C_i = pk_i$.

Forgery After the signature queries, \mathcal{A} outputs a forged signature on a new message m^* as $\sigma^* = (A^*, B^*, C^*)$.

If $C^* \neq C_i$ for $1 \leq i \leq q$, then \mathcal{A}' can output a forged Π' signature as $\sigma = A^*$ on a new message C^* and break the signature scheme Π' . Otherwise, \mathcal{A}' aborts.

If $b_{code} = 2$, we construct an algorithm \mathcal{A}' to break Π' as follows:

Simulation of Key Generation. \mathcal{A}' randomly generates $(pk, sk) \leftarrow \text{Gen}'(1^k)$ of Π' . \mathcal{A}' then sets Π' 's key pair as (pk, sk) and sends the public key pk to \mathcal{A} . \mathcal{A}' also chooses a random $\kappa \in [1, q]$ and keeps it secret.

Simulation of Signing Oracle. \mathcal{A} queries the signing oracle of Π on messages m_i adaptively for $1 \leq i \leq q$. \mathcal{A}' answers the signature query as follows: if $i \neq \kappa$, \mathcal{A}' first invokes $\text{Gen}'(1^k)$ and gets key pair $(pk_i, sk_i) \leftarrow \text{Gen}'(1^k)$. Then, it returns the simulated signature on messages m_i as $\sigma_i=(A_i, B_i, C_i)$, where $A_i=\text{Sign}'(sk, pk_i)$, $B_i = \text{Sign}'(sk_i, m_i)$, $C_i = pk_i$. Otherwise, if $i = \kappa$, \mathcal{A}' sends m_i to the challenger

for signature of Π' , and gets the challenge public key pk^* of Π' and signature $\text{Sign}'(sk^*, m_i)$ of m_i with respect to pk^* . Then \mathcal{A}' answers the signature query as $\sigma_i = (A_i, B_i, C_i)$, where $A_i = \text{Sign}'(sk, pk^*)$, $B_i = \text{Sign}'(sk^*, m_i)$ and $C_i = pk^*$.

Forgery. After the signature queries, \mathcal{A} outputs a forged signature on a new message m^* as $\sigma^* = (A^*, B^*, C^*)$, where $C^* = C_i$ for some $1 \leq i \leq q$.

If $i \neq \kappa$, \mathcal{A}' aborts and fails. Otherwise, if $i = \kappa$, then $c^* = pk^*$. This implies that \mathcal{A}' can output a forged signature B^* on a new message m^* with respect to pk^* and break the signature scheme Π' .

We define two events, E_1 and E_2 , which denotes type 1 forgery and type 2 forgery occurs, respectively. As $\text{prob}[E_1] + \text{prob}[E_2] = \text{prob}[\mathcal{A} \text{ wins}]$. Since \mathcal{A} wins with probability ε , it follows that one of the two events occurs with probability at least $\varepsilon/2$. It is easy to see that the success probability of \mathcal{A}' under the conditions that event E_1 occurs is $\frac{1}{2} \cdot \text{prob}[E_1]$. In the type 2 forgery simulation, success guess of γ is $\frac{1}{q}$. So the success probability of \mathcal{A}' under the conditions that event E_2 occurs is $\frac{1}{2q} \text{prob}[E_2]$. Therefore, if \mathcal{A} wins with probability ε , the signature scheme Π' with probability at least $\frac{\varepsilon}{2q}$. \square

Appendix B: Proof of Theorem 4

Proof. Given any adversary \mathcal{A} attacking Π in an adaptive chosen message attack, we construct an adversary \mathcal{A}' breaking Π' in weak chosen message attacks. After given public key pk of Π , \mathcal{A} queries the signing oracle of Π on messages m_i adaptively and gets q signatures $\sigma_i = (A_i, B_i, C_i)$ for $1 \leq i \leq q$. After the signature queries, \mathcal{A} outputs a forged signature on a new m^* as $\sigma^* = (A^*, B^*, C^*)$.

There are also two types of forgeries:

Type 1 forgery: $C^* \neq C_i$ for $1 \leq i \leq q$.

Type 2 forgery: $C^* = C_i$ for some i , $1 \leq i \leq q$.

The reduction works differently for each forger type. Therefore, initially \mathcal{A}' will choose a random bit $b_{code} \in \{1, 2\}$ that indicates its guess for the type of forger that \mathcal{A} will emulate. The simulation proceeds differently for each b_{code} .

If $b_{code} = 1$, we construct an algorithm \mathcal{A}' to break Π' as follows:

Simulation of Key Generation. \mathcal{A}' first invokes $\text{Gen}'(1^k)$ and gets key pair $(pk_2, sk_2) \leftarrow \text{Gen}'(1^k)$. Then \mathcal{A}' chooses q random values m'_1, \dots, m'_q (Assume \mathcal{A} makes at most q queries to signing oracle), and sends the q values m'_i , for $1 \leq i \leq q$, to challenger for signature queries of Π' before the parameters publication of Π' . Then \mathcal{A}' gets its challenge public key \overline{pk} of Π' and q signatures $\sigma'_i = \text{Sign}'(\overline{sk}, m'_i)$ on the q messages m'_i , with respect to \overline{pk} , for $1 \leq i \leq q$.

Then \mathcal{A}' sets the public key of Π as $pk = (pk_1, pk_2)$, where $pk_1 = \overline{pk}$, and sends the public key pk to the adversary \mathcal{A} .

Simulation of Signing Oracle. \mathcal{A} then queries the signing oracle of Π on messages m_i adaptively for $1 \leq i \leq q$. \mathcal{A}' answers the signature query as follows:

- From the first property of the given relation \mathcal{R} , \mathcal{A}' could compute m_i'' such that $((m_i', m_i''), m_i) \in \mathcal{R}$;
- Then, it computes $B_i = \text{Sign}'(sk_2, m_i'')$;
- Finally, outputs the signature $\sigma_i = (A_i, B_i, C_i)$, where $A_i = \sigma_i'$ from challenger, and $C_i = m_i'$.

Forgery. After the signature queries, \mathcal{A} outputs a forged signature on a new message m^* as $\sigma^* = (A^*, B^*, C^*)$.

Because $m^* \neq m_i$ for $1 \leq i \leq q$, then if $C^* = m_i'$ for some i , \mathcal{A}' aborts and fails. Otherwise, \mathcal{A}' can output a forged Π' signature as $\sigma = A^*$ of a new message C^* and break the signature scheme Π' , with the challenge public key \overline{pk} .

If $b_{code} = 2$, we construct an algorithm \mathcal{A}' to break Π' in another way:

Simulation of Key Generation. \mathcal{A}' randomly generates $(pk_1, sk_1) \leftarrow \text{Gen}'(1^k)$ of Π' . Then \mathcal{A}' chooses q random values m_1'', \dots, m_q'' (Assume that \mathcal{A} makes at most q queries to signing oracle), and sends the q values m_i'' , for $1 \leq i \leq q$, to challenger for signature queries of Π' before the parameters publication of Π' . Then \mathcal{A}' gets its challenge public key \overline{pk} of Π' and q signatures $\sigma_i' = \text{Sign}'(\overline{sk}, m_i'')$ of the q messages m_i'' , with respect to \overline{pk} , for $1 \leq i \leq q$.

Then \mathcal{A}' sets the public key of Π as $pk = (pk_1, pk_2)$, where $pk_2 = \overline{pk}$, and sends the public key pk to the adversary \mathcal{A} .

Simulation of Signing Oracle. \mathcal{A} then queries the signing oracle of Π on messages m_i adaptively for $1 \leq i \leq q$. \mathcal{A}' answers the signature query as follows: First, from the first property of relation \mathcal{R} , \mathcal{A}' computes m_i' such that $((m_i', m_i''), m_i) \in \mathcal{R}$. Then, \mathcal{A}' outputs simulated signature on message m_i as $\sigma_i = (A_i, B_i, C_i)$, where $A_i = \text{Sign}'(sk_1, m_i')$, $B_i = \sigma_i'$, $C_i = m_i'$.

Forgery. After the signature queries, \mathcal{A} outputs a forged signature on a new message m^* as $\sigma^* = (A^*, B^*, C^*)$. By using the first property of relation \mathcal{R} again, \mathcal{A}' could compute $m^{*''}$ from m^* and C^* , such that $((C^*, m^{*''}), m^*) \in \mathcal{R}$.

Recall that in this kind of forgery, $C^* = C_i$ for some i . Because $m^* \neq m_i$ for $1 \leq i \leq q$, and m_i', m_i'' are chosen randomly by the simulator, we have $m^{*''} \neq m_i''$ from the second property of the defined relation \mathcal{R} . This proof, in fact, shows that the signature scheme prevents the attack from the adversary that just combine the first part in one signature for message M and the second part in the other signature for message M' .

So, \mathcal{A}' can output a forged Π' signature as $\sigma = A^*$ on a new message $m^{*''}$ and break the signature scheme Π' , with respect to the challenge public key \overline{pk} . □

New Differential-Algebraic Attacks and Reparametrization of Rainbow

Jintai Ding¹, Bo-Yin Yang², Chia-Hsin Owen Chen², Ming-Shing Chen²,
and Chen-Mou Cheng³

¹ Dept. of Mathematical Sciences, University of Cincinnati, USA
ding@math.uc.edu

² IIS, Academia Sinica, Taiwan

{bbyang,owenhsin,mschen}@iis.sinica.edu.tw

³ Dept. of Elec. Eng., Nat'l Taiwan University, Taiwan
ccheng@cc.ee.ntu.edu.tw

Abstract. A recently proposed class of multivariate Public-Key Cryptosystems, the Rainbow-Like Digital Signature Schemes, in which successive sets of central variables are obtained from previous ones by solving linear equations, seem to lead to efficient schemes (TTS, TRMS, and Rainbow) that perform well on systems of low computational resources. Recently SFLASH (C^{*-}) was broken by Dubois, Fouque, Shamir, and Stern via a differential attack. In this paper, we exhibit similar algebraic and differential attacks, that will reduce published Rainbow-like schemes below their security levels. We will also discuss how parameters for Rainbow and TTS schemes should be chosen for practical applications.

Keywords: rank, differential attack, algebraic attack, oil-and-vinegar.

Note: Up-to-date version will be available at eprint.iacr.org/2008/108

1 Outline

Multivariate Public-Key Cryptosystems (MPKCs, or trapdoor \mathcal{MQ} schemes) are cryptosystems for which the public key is a set of polynomials $\mathcal{P} = (p_1, \dots, p_m)$ in variables $\mathbf{x} = (x_1, \dots, x_n)$ where all variables and coefficients are in $\mathbb{K} = \text{GF}(q)$. In practice this is always accomplished via

$$\mathcal{P} : \mathbf{w} = (w_1, \dots, w_n) \in \mathbb{K}^n \xrightarrow{S} \mathbf{x} = M_S \mathbf{w} + \mathbf{c}_S \xrightarrow{Q} \mathbf{y} \xrightarrow{T} \mathbf{z} = M_T \mathbf{y} + \mathbf{c}_T = (z_1, \dots, z_m) \in \mathbb{K}^m$$

In any given scheme, the *central map* Q belongs to a certain class of quadratic maps whose inverse can be computed relatively easily. The maps S, T are affine. The polynomials giving y_i in \mathbf{x} are called the central polynomials, and the x_j are called the central variables.

In 1999, the Unbalanced Oil-and-Vinegar multivariate structure is proposed by Patarin *et al* [16]. Lately the Rainbow class of signatures [7, 25, 20], based on

repeated applications of the Unbalanced Oil-and-Vinegar principle, shows some promise on systems of low computational resources.

Given that the well-known C^{*-} class of signature schemes including SFLASH was broken by differential attacks [8], we examine similar attacks on Rainbow, with the following conclusions:

- Differentials improve on the High-Rank attacks on Rainbow-like systems.
- Differentials also helps with randomized brute-force searches for S and T .
- We can assess how Rainbow-like schemes needs to be amended in view of recent developments.
- The results are in line with experiments run on small scale systems.

In Sec. 2 we recap Rainbow-like multivariates and what is known about the security of MPKC before the appearance of Rainbow in Sec. 3. In Sec. 4 we describe the new differential attack, which is related to the high-rank attack, and in Sec. 5 we present new paramters for Rainbow construction, we tabulate what we know about the security of Rainbow-like schemes, in particular, the security against the two new recent attacks specially targeted against the Rainbow schemes, and we design schemes with new parameters for practical applications. Finally, in Sec. 6 we present the conclusion.

2 Rainbow-Like Multivariate Signatures

We characterize a Rainbow type PKC with u stages:

- The segment structure is given by a sequence $0 < v_1 < v_2 < \dots < v_{u+1} = n$. For $l = 1, \dots, u + 1$, set $S_l := \{1, 2, \dots, v_l\}$ so that $|S_l| = v_l$ and $S_0 \subset S_1 \subset \dots \subset S_{u+1} = S$.
- Denote by $o_l := v_{l+1} - v_l$ and $O_l := S_{l+1} \setminus S_l$ (i.e., $v_l < k \leq v_{l+1}$ if $k \in O_l$) for $l = 1 \dots u$. The central map $\mathcal{Q} : \mathbf{x} = (x_1, \dots, x_n) \mapsto \mathbf{y} = (y_{v_1+1}, \dots, y_n)$, where each $y_i := q_i(\mathbf{x})$ is a quadratic polynomial in \mathbf{x} of the following form

$$q_k = \sum_{i < j \leq v_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \leq v_l < j < v_{l+1}} \alpha_{ij}^{(k)} x_i x_j + \sum_{i < v_{l+1}} \beta_i^{(k)} x_i.$$

In every q_k , $k \in O_l$, there is no cross-term $x_i x_j$ where both i and j are in O_l at all. So given all the y_i with $v_l < i \leq v_{l+1}$, and all the x_j with $j \leq v_l$, we can compute $x_{v_l+1}, \dots, x_{v_{l+1}}$.

- To expedite computations, some coefficients α_{ijk} 's may be fixed (e.g., set to zero), chosen at random (and included in the private key), or be interrelated in a predetermined manner.
- To invert \mathcal{Q} , determine (usu. at random) x_1, \dots, x_{v_1} , i.e., all x_k , $k \in S_1$. From the components of \mathbf{y} that corresponds to the polynomials $q_{v_1+1}, \dots, q_{v_2}$, we obtain a set of o_1 equations in the variables x_k , ($k \in O_1$). We may repeat the process to find all remaining variables.

In this form, we can see that Rainbow can only be a signature scheme. We can see a good example of what can go wrong in [15] if we try to construct an encryption scheme, where the initial vinegar variables is determined through an initial block of equations.

Example 1. enTTS(20,28) of [25] has structure (8, 9, 1, 1, 9) and this central map:

$$\begin{aligned}
 y_i &= x_i + \sum_{j=1}^7 p_{ij}x_jx_{8+(i+j \bmod 9)}, \quad i = 8 \cdots 16; \\
 y_{17} &= x_{17} + p_{17,1}x_1x_6 + p_{17,2}x_2x_5 + p_{17,3}x_3x_4 \\
 &\quad + p_{17,4}x_9x_{16} + p_{17,5}x_{10}x_{15} + p_{17,6}x_{11}x_{14} + p_{17,7}x_{12}x_{13}; \\
 y_{18} &= x_{18} + p_{18,1}x_2x_7 + p_{18,2}x_3x_6 + p_{18,3}x_4x_5 \\
 &\quad + p_{18,4}x_{10}x_{17} + p_{18,5}x_{11}x_{16} + p_{18,6}x_{12}x_{15} + p_{18,7}x_{13}x_{14}; \\
 y_i &= x_i + p_{i,0}x_{i-11}x_{i-9} + \sum_{j=19}^{i-1} p_{i,j-18} x_{2(i-j)-(i \bmod 2)} x_j + p_{i,i-18}x_0x_i \\
 &\quad + \sum_{j=i+1}^{27} p_{i,j-18} x_{i-j+19} x_j, \quad i = 19 \cdots 27.
 \end{aligned} \tag{1}$$

If x_0, \dots, x_7 is decided, one can solve first for x_8, \dots, x_{16} , then x_{17}, x_{18} , then x_{19}, \dots, x_{27} . Note: x_0 does not appear until the last block, which will be significant later.

Example 2. The proposed Rainbow scheme in [7] is an essentially generic stage-wise UOV construction with layers (6, 6, 5, 5, 11). The first six central equations is a generic UOV construction with six vinegar (x_1, \dots, x_6) and six oil (x_7, \dots, x_{12}) variables; the next five has 12 vinegars and 5 oils (x_{13}, \dots, x_{17}); the next five has 17 vinegars and 5 oils (x_{18}, \dots, x_{22}), and the last 11 has 22 vinegars and 11 oils (x_{23}, \dots, x_{33}).

Rainbow schemes where most of the crossterm coefficients $\alpha_{ij}^{(k)}$ are zero are said to be TTS instances. TTS schemes have a relatively small private key and even better efficiency, but may be exposed to additional risks. Regardless, the same techniques that we shall describe below are security concerns for all schemes of the rainbow type including TTS, TRMS, and Rainbow [25,20,7].

3 The Security of Multivariates and Prior Attacks

The name of the class came from the ‘‘Multivariate Quadratics’’ problem:

Problem MQ: Solve the system $p_1 = p_2 = \dots = p_m = 0$, where each p_i is a quadratic polynomial in $\mathbf{x} = (x_1, \dots, x_n)$ and coefficients and variables are in $\mathbb{K} = \text{GF}(q)$.

Generic MQ is NP-hard [12], and consensus pegs it as a difficult problem to solve even probabilistically. However, to use MQ as the underlying hard problem in a PKC, one need a trapdoor built into the public map \mathcal{P} . So the security of the cryptosystem also depends on the following:

Problem EIP: (Extended Isomorphism of Polynomials) Given a class of central maps \mathfrak{C} and a map \mathcal{P} expressible as $\mathcal{P} = T \circ \mathcal{Q} \circ S$, where $\mathcal{Q} \in \mathfrak{C}$, and S, T are affine, make such a decomposition.

There are two interesting twists here:

- If \mathcal{Q} is constant, this is known as the IP problem. J.-C. Faugère showed that in some cases simple IP is not NP-hard at Eurocrypt 2006 [11].
- The EIP problem where \mathcal{C} is the set of homogeneous quadratic maps is easy [13]. Equivalently, if \mathcal{Q} is homogeneous (e.g., as in SFLASH= C^{*-}) we can set $\mathbf{c}_S = \mathbf{c}_T = 0$.

If \mathcal{Q} fundamentally involves a map in a field $\mathbb{L} = \mathbb{K}^k$ that is of a size significantly bigger than \mathbb{K} , we call the scheme “big field” or “dual field”. This order includes derivatives of Matsumoto-Imai (C^*) and Hidden Field Equations. Otherwise we call the scheme a “true multivariate” (sometimes “single field”). This includes the Unbalanced Oil-and-Vinegar and stagewise triangular structures.

One of the biggest concerns of multivariate cryptography is the lack of provable security results. Today security in MPKC is still very much *ad hoc*. Proposed schemes are evaluated against known attacks security estimates obtained for various parameters. The designers then tries to juggle the system parameters so as to have some requisite security level under every known attack.

With that, we list the standard attacks known for MPKCs today:

1. Rank (or Low Rank, MinRank) attack, which finds a central equation with least rank [25].

$$C_{\text{low rank}} \approx \left[q^{r \lceil m/n \rceil} m(n^2/2 - m^2/6) / \mu \right] \mathbf{m}.$$

Here as below, the unit \mathbf{m} is a multiplications in \mathbb{K} , and r is that lowest rank (“MinRank”, [14]). μ is the number of linear combinations of central equations [25] at that minimal rank.

2. Dual Rank (or High Rank) attack [5,14], which finds a variable appearing the fewest number of times in a central equation cross-term. If this least number is s , [25] gives

$$C_{\text{high rank}} \approx \left[q^s n^3 / 6 \right] \mathbf{m}.$$

3. Oil-and-Vinegar Separation [22,16,17], which finds an Oil subspace that is sufficiently large (estimates as corrected in [25]).

$$C_{\text{UOV}} \approx \left[q^{n-2o-1} o^4 + (\text{some residual term bounded by } o^3 q^{m-o} / 3) \right] \mathbf{m}.$$

o is the max. *oil set* size, i.e., there is a set of o central variables which are never multiplied together in the central equations, and no more.

4. Trying for a direct solution (i.e., going for the \mathcal{MQ} as opposed to the EIP or “structural” problem). Best known methods are the Lazard-Faugère family of solvers (the Gröbner Bases methods \mathbf{F}_4 - \mathbf{F}_5 or XL) whose complexities [6,9,10,24] are very hard to evaluate; some recent asymptotic formulas can be found in [1,2,24].

4 New Differential Attacks

One key point of our new attack is to use the differentials (first used, as far as we know, with MPKC in [18] and recently to break SFLASH [8]).

Given the public key of a MPKC, which we denote as $\mathcal{P}(\mathbf{x})$, a set of quadratic polynomials, its differential $D\mathcal{P}(\mathbf{x})$ is defined as

$$D\mathcal{P}(\mathbf{x}) = \mathcal{P}(\mathbf{x} + \mathbf{c}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{c}),$$

a set of linear functions in \mathbf{x} .

The key is to use the hidden structures in the differential to attack the cryptosystem. The observation is that the differential can be used to improve the old high-rank attack when there are too many variables that don't appear in the final block of equations (for y_i , where $i \in O_u$). First, we will reformulate an existing attack in terms of the differentials.

Let H_i be the symmetric matrix corresponding to the quadratic part of $z_i(\mathbf{w})$. Without loss of generality, we may let the fewest number of appearances of all variables in the cross-terms of the central equations be the last variable x_n appearing s times.

Algorithm 0 (High or Dual Rank Attack). *as described by Goubin-Courtois and Yang-Chen [14, 25]:*

1. Compute the differential $\mathcal{P}(\mathbf{x} + \mathbf{c}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{c})$ and take its j -th component (which is bilinear in \mathbf{x} and \mathbf{c}) as $\mathbf{c}^T H_j \mathbf{x}$. H_k is representing the quadratic crossterms in the k -th polynomial of the public key. Note that the H_i are always symmetric and if $\text{char}\mathbb{K} = 2$, and $\mathbf{x}^T H_i \mathbf{x} = 0$.
2. Form an arbitrary linear combination $H = \sum_i \alpha_i H_i$. Find $V = \ker H$.
3. When $\dim V = 1$, set $(\sum_j \lambda_j H_j)V = \{\mathbf{0}\}$ and check if the solution set \hat{V} of the (λ_i) form a subspace dimension $m - s$. Note: Since a matrix in $K^{n \times n}$ can have at most n different eigenvalues, less than n/q of the time we would need to do this.
4. With probability q^{-s} we have $V = U = \{\mathbf{x} : x_1 = \dots = x_{v_u} = 0\}$.

As each trial run consists of running an elimination and some testing, we can realistically do this with $\sim \left(sn^2 + \frac{n^3}{6} \right) q^s$ field multiplications, by taking linear combinations from only $(s+1)$ of the matrices H_i and hope not to get too unlucky. An upper bound is $\left[mn^2 + \frac{n^3}{6} + \frac{n}{q}(m^3/3 + mn^2) \right] q^s$.

The above formulation of the high rank attack is designed to defeat “plus”-modified Triangular systems. We first present some notations before describing how we can improve this attack further:

Let P_l be the linear space of quadratic polynomials spanned by polynomials of the form

$$\sum_{i \in O_l, j \in S_l} \alpha_{i,j} x_i x_j + \sum_{i,j \in S_l} \alpha_{i,j} x_i x_j + \sum_{i \in S_{l+1}} \beta_i x_i + \eta$$

We can see that these are Oil and Vinegar type of polynomials such that $x_i, i \in O_l$ are the Oil variables and $x_i, i \in S_l$ are the Vinegar variables. We call $x_i, i \in O_l$ an l -th layer Oil variable and $x_i, i \in S_l$ an l -th layer Vinegar variable. We call any polynomial in P_l an l -th layer Oil and Vinegar polynomial. Clearly we have $P_i \subset P_j$ for $i < j$. Let W_i be the space of linear functions of variables x_1, \dots, x_{v_i} . Then we have

$$W_1 \subset P_1 \subset W_2 \subset P_2 \cdots \subset W_u \subset P_u \subset W_{u+1}.$$

Now we present the new attack:

Algorithm 1. *The Improved High-Rank Attack using differentials:*

1. Pick random $\mathbf{c}, \mathbf{c}' \in \mathbb{K}^n$, compute $\mathcal{P}(\mathbf{w} + \mathbf{c}) - \mathcal{P}(\mathbf{w}) - \mathcal{P}(\mathbf{c})$, and we will denote its components as (t_1, t_2, \dots, t_m) . Similarly we compute $(t'_1, t'_2, \dots, t'_m) = \mathcal{P}(\mathbf{w} + \mathbf{c}') - \mathcal{P}(\mathbf{w}) - \mathcal{P}(\mathbf{c}')$, then

$$U = \text{span}(t_1, t_2, \dots, t_m) \cap \text{span}(t'_1, t'_2, \dots, t'_m).$$

2. Guess at a linear form $f \in U$; find coefficients a_i and a'_i such that $f = \sum a_i t_i = \sum a'_i t'_i$.
3. Use a_i and a'_i as the guessed α_i in the High Rank Attack (Algorithm 1) above.

Proposition 1. *The expected complexity of Algorithm 1 is $\sim q^d$ (cubic-time elimination) where (the last block of equations is the ones whose solutions gives O_u)*

$$d \leq s - [\# \text{ vars appearing in crossterms only in the last block}]. \tag{2}$$

Proof. Let

$$F = (F_1, \dots, F_m) = Q \circ S$$

be the mapping from $\mathbf{x} \mapsto \mathbf{z}$. Let

$$F(\mathbf{x} + \mathbf{b}) - F(\mathbf{x}) - F(\mathbf{b}) := G = (G_1, G_2, \dots, G_n),$$

where $\mathbf{b} = (b_1, b_2, \dots, b_i, \dots, b_n)$ is randomly chosen. Pick another \mathbf{b}' and form

$$H = (H_1, \dots, H_n) = F(\mathbf{x} + \mathbf{b}') - F(\mathbf{x}) - F(\mathbf{b}'),$$

then

1. if $i \in O_j$, then $G_i, H_i \in W_{j+1}$;
2. $\overline{W}_{j+1} := \text{span}\{G_i\}_{i \in O_j} \subset W_{j+1}$, and similarly $\widehat{W}_{j+1} := \text{span}\{H_i\}_{i \in O_j} \subset W_{j+1}$;
3. $\overline{W}_2 \subset \dots \subset \overline{W}_{u+1}$ and $\widehat{W}_2 \subset \dots \subset \widehat{W}_{u+1}$.

Clearly $(\widehat{W}_u \cap \overline{W}_u) \subset (\widehat{W}_{u+1} \cap \overline{W}_{u+1})$, and we observe that: if the dimensions of the two subspaces differ by d , then we can break the system with $\propto q^d$ (one guess) computations.

How so? Because the relationship between \mathcal{P} and F , is the same as that between the \mathbf{w} -space and \mathbf{x} -space, i.e., the linear transformation S . So there is a $1\text{-in-}q^d$ chance that both $\sum a_i z_i$ and $\sum a'_i z_i$ correspond to a linear form in W_u . The odds are now decided by q^{-d} instead of q^{-s} . In a Rainbow-like system, $s = o_u = n - v_u$. For Alg. 1 to be worthwhile, we must show that $d \leq s$.

In fact, it is not so hard to describe how to determine d . \overline{W}_{u+1} and \widehat{W}_{u+1} are two m -dimensional subspaces in the n -dimensional vector space W_{u+1} . Most of the time they intersect in a $2m - n$ dimensional subspace, hence

$$\dim \overline{W}_u = \dim \widehat{W}_u = m - o_u$$

which equals the number of variables appearing in cross-terms in equations not of the final block, which is equivalent to Eq. 2

Example 3. Consider enTTS(20,28) as in Eq. 1. Here $\dim(\overline{W}_{u+1} \cup \widehat{W}_{u+1}) = 20 + 20 - 28 = 12$, while $\dim(\overline{W}_u \cup \widehat{W}_u) = 11 + 11 - 17 = 5$. Therefore we need only $\sim 2^{56}$ instead of 2^{72} guesses, which is a speed increase of $2^{16} \times$ over Algorithm 1. Since each guess takes about 2^8 time units (standard is to use time of a 3DES block encryption, between 2^6 to 2^8 multiplications), this gives complexity 2^{64} instead of 2^{80} , too weak to be “strong” crypto.

What went wrong? Generically $\dim W_u = n - o_u$ and the intersection is of dimension $2(m - o_u) - (n - o_u) = 2m - n - o_u$, making $d = (2m - n) - (2m - n - o_u) = o_u = s$. *The lesson: watch out for variable not in the final oil set that does not occur prior to the last block of equations.* In enTTS(20,28), x_0 and x_{18} did not appear in any earlier equations than the final block.

4.1 Experimentation with Mini-versions

We experimented in smaller fields with three different schemes: Rainbow (6,6,5,5, 11), the enTTS(20,28) scheme above, and its miniaturized sister version enTTS (16,22) [structure (6,7,1,1,7)].

Table 1. Timing (sec) on 16 of 3GHz P4 machines guessing in parallel

Scheme	Structure	q	Alg. 0	Alg. 1	ratio
EnTTS(20,28)	(8,9,1,1,9)	8	93	4.4	0.047
EnTTS(20,28)	(8,9,1,1,9)	16	42435	496	0.012
EnTTS(16,22)	(6,7,1,1,7)	16	102	3.5	0.034
Rainbow [7]	(6,6,5,5,11)	8	8454	17123	2.028

The results are fairly constant over many tests [except the enTTS(20,28) test which we only ran a few times]. Clearly, not having all vinegar variables of the last segment appearing previously in cross-terms is a big minus. Rainbow (6,6,5,5,11) does not have the same problem and Algorithm 1 is no improvement of the High Rank Attack against it.

5 New Rainbow Parameters for Practical Applications

For practical applications, we will propose the following Rainbow Structures.

1. $(20, 10, 4, 10)$, where the public key has 44 variables and 24 polynomials.
2. $(18, 12, 12)$, where the public key has 42 variables and 24 polynomials.
3. $(20, 14, 14)$, where the public key has 48 variables and 28 polynomials.

We will first formalize a twist on the regular Rainbow construction, which is somewhat more general. In the previous constructions, in each new layer, previously appeared variables will only be Vinegar variables, the new variables appearing only as Oil variables. We can also consider adding new Vinegar variables as we add Oil variables. This also implies that in the signing process, we guess at the new vinegar variables as they appear, while in the previous Rainbow construction, we only guess the Vinegar variables in the first layer once. In this case, we can also write for each layer two parameters, (v'_i, o_i) , where the v'_i counts the new vinegar variables we introduce. In this layer, we will have $v_i + v'_i$ Vinegar variables (where v_i counts the number of all previous appearing variables) and o_i the number of Oil variables.

If all the v'_i are zero, this is precisely the original Rainbow construction. We might call this new construction the extended Oil-Vinegar construction. From the viewpoint of the attacker we can see this as a specialization of the Rainbow construction, since the new vinegar variables might as well have been part of the initial block of vinegar variables, but simply never have been used before. However, it is different in an operative sense, in that if we use the new vinegar variables properly, we could always find a signature, as implicitly used in TTS constructions earlier.

So, in this language, we would propose scheme: $((15,10), (4, 4), (1, 10)), ((17, 12), (1, 12))$, and $((19, 14), (1, 14))$.

For these new schemes, we could also choose to use the generic sparse polynomials or special sparse polynomials as in the case of TTS [23]. For generic sparse polynomials, we think it is a good idea to choose $3L_i$ terms for each layer, where L_i is the sum of number of Oil and vinegar variables in each layer.

For these new schemes, we need to take into two new recent special attacks against Rainbow.

5.1 The Reconciliation Attack

In the following attack we attempt to find a sequence of change of basis that let us invert the public map. In this sense it can be considered an improved brute force attack.

Suppose we have an oil-and-vinegar structure, then the quadratic part of each component q_i in the central map from \mathbf{x} to \mathbf{y} , when expressed as a symmetric matrix, looks like

$$M_i := \left[\begin{array}{ccc|ccc} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & \alpha_{1,v+1}^{(i)} & \cdots & \alpha_{1n}^{(i)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & \alpha_{v,v+1}^{(i)} & \cdots & \alpha_{vn}^{(i)} \\ \hline \alpha_{v+1,1}^{(i)} & \cdots & \alpha_{v+1,v}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1}^{(i)} & \cdots & \alpha_{nv}^{(i)} & 0 & \cdots & 0 \end{array} \right] \tag{3}$$

First, no matter what M_T is, it won't change the basic shape, so we let T be the identity map for the moment. What can S be like? Suppose we pick M_S as totally random, most often (see below) it decompose to

$$M_S := \begin{bmatrix} *_{v \times v} & *_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix} = \begin{bmatrix} 1_{v \times v} & *_{v \times o} \\ 0_{o \times v} & 1_{o \times o} \end{bmatrix} \begin{bmatrix} *_{v \times v} & 0_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix} \tag{4}$$

where 1 means identity matrix, 0 means just zeros and * means random or anything. In fact, this decomposition always hold unless the lower-right $o \times o$ submatrix is singular. It should be clear that the $\begin{bmatrix} *_{v \times v} & 0_{v \times o} \\ *_{o \times v} & *_{o \times o} \end{bmatrix}$ portion of M_S , as a coordinate change leaves the M_i 's with the same shape. That is, if we can find the correct $\begin{bmatrix} 1_{v \times v} & *_{v \times o} \\ 0_{o \times v} & 1_{o \times o} \end{bmatrix}$ portion and perform the basis change in reverse, we will again make the resulting public map into the same form (all zeroes on the lower right) and be easily inverted. Hence, no more security at all. More about this phenomenon ("equivalent keys") in MPKCs can be seen in, say, [23].

Let this essential portion of M_S that we wish to recreate be P , that is, the linear transformation $\mathbf{w} \mapsto \mathbf{x} = P\mathbf{w}$ will create all zeroes on the lower right. We can decompose this P into a product of $P := P_{v+1}P_{v+2} \cdots P_n$, where each of the matrices look like

$$P_n = 1_n + \begin{bmatrix} 0 \cdots 0 & a_1 \\ 0 \cdots 0 & a_2 \\ \vdots & \vdots \\ 0 \cdots 0 & a_v \\ \hline 0 \cdots 0 & 0 \\ \vdots & \vdots \\ 0 \cdots 0 & 0 \end{bmatrix} ; \quad P_{n-1} = 1_n + \begin{bmatrix} 0 \cdots 0 & a'_1 & | & 0 \\ 0 \cdots 0 & a'_2 & | & 0 \\ \vdots & \vdots & | & \vdots \\ 0 \cdots 0 & a'_v & | & 0 \\ \hline 0 \cdots 0 & 0 & | & 0 \\ \vdots & \vdots & | & \vdots \\ 0 \cdots 0 & 0 & | & 0 \end{bmatrix} ; \cdots$$

Indeed, the multiplication is actually commutative among the various P_i 's. *Suppose, then, that we start with the differential matrices H_i and simultaneously transform them to make their lower-right corner a square of 0's using exactly such P_i 's.*

Algorithm 2 (UOV Reconciliation). *The following gives the Reconciliation Attack against a UOV scheme with o oil and $v = n - o$ vinegar variables (which has the smaller indices):*

1. Perform basis change $w_i := w'_i - \lambda_i w'_n$ for $i = 1 \dots v$, $w_i = w'_i$ for $i = v + 1 \dots n$. Evaluate \mathbf{z} in \mathbf{w}' .
2. Let all coefficients of $(w'_n)^2$ be zero and solve for the λ_i . We may use any method such as $\mathbf{F}_4/\mathbf{F}_5$ or FXL. There will be m equations in v unknowns.
3. Repeat the process to find P_{n-1} . Now we set $w'_i := w''_i - \lambda_i w''_{n-1}$ for $i = 1 \dots v$, and set every $(w''_{n-1})^2$ and $w''_n w''_{n-1}$ term to zero (i.e., more equations in the system) after making the substitution. This time there are $2m$ equations in v unknowns.
4. Continue similarly to find P_{n-2}, \dots, P_{v+1} with more and more equations.

Given what we know about system-solving today, we can expect the complexity to be determined in solving the initial system. Hence, if $v < m$, solving m equations in v variables will be easier than m equations in n equations, and we achieve a simplification.

Proposition 2. *The Reconciliation Attack works with probability $\approx \left(1 - \frac{1}{q-1}\right)$.*

Proof (Sketch). Provided that lower-right $o \times o$ submatrix of M_S is non-singular, we can see that the construction of P_n will eliminate the quadratic term in the last variable. P_{n-1} will eliminate all quadratic terms in the last two variables, and so on, and each sequential construction will not disturb the structure built by the prior transformations. The number of nonsingular $k \times k$ matrices in over $\text{GF}(q)$ is $(q^k - 1)(q^k - q)(q^k - q^2) \dots (q^k - q^{k-1})$, because the first row has 1 possibility to be zero, the second row q possibilities to be a multiple of the first, the third row q^2 possibilities to be dependent on the first two, etc., so the chance that the above attack works is roughly

$$\left(1 - \frac{1}{q}\right) \left(1 - \frac{1}{q^2}\right) \dots \left(1 - \frac{1}{q^k}\right) > 1 - \left(\frac{1}{q} + \frac{1}{q^2} + \dots + \frac{1}{q^k}\right) > 1 - \frac{1}{q-1}.$$

Here we will use formulas from [26] for all our estimates as shown below.

Example 4. We attack enTTS(20,28) as in Eq. [1]. Originally we must solve a 20-equation, 20-variable (we can guess 8 out of the original 28) MQ system. With $v_u = 19$, the rate-determining step of the Reconciliation Attack is a 20-equation, 19-variable system. This is easier by a factor of exactly 2^8 if we are using FXL or FF_4 [24, 1], since we will guess exactly one fewer variable.

Since we expect a direct attack on enTTS(20,28) to have $\sim 2^{72}$ complexity, Alg. [2] should take $\sim 2^{64}$. The construction process and odds as given above have been tested and verified on miniature versions (cf. [25]) of TTS schemes such as enTTS(16,22) as well as other Rainbow-like instances.

Example 5. TRMS [20] can be reduced to 2^{64} via the same attack (a faster attack given below) because it has rainbow layer parameters of (8, 6, 2, 3, 9), with a last block of the same size as TTS.

Example 6. We implemented enTTS(16,22) over $\text{GF}(128)$, the initial system has 16 equations and $22 - 7 = 15$ variables. We ran FXL with Wiedemann solver

(as in [26]) with one fixed variable on an assembly of machines with 128 total P4 cores at 3.0GHz, each guessing 1 value out of 128. Here $D = 8$ [24], and the number of monomials is $T = 319770$, with a total of 73799040 terms which took only 288MB of storage at every core. Solving a system known to have a solution should take around $3(T^2n(n + 3)/2) \approx 2^{45}$ multiplications, which at about 16 cycles a multiplication about 2.0×10^4 seconds, but we discovered that there is guesswork in generating a system, so we dare not run more than one value on a given CPU.

In practice we were not so unlucky and were able to solve 15 variables in 16 equations in GF(128) in what was in fact closer about 3 days, probably due to non-optimal programming. After that, solving the remaining systems is a piece of cake [real CPU time estimated at less than two hours], and we can then decompose an enTTS(16,22) instance.

Example 7. We now attack the proposed Rainbow instance in [7]. Since $v_u = 22 < m = 26$, solving this one is significantly easier: using \mathbf{FF}_5 [24], the expected time use is 2^{56} (3DES blocks) instead of 2^{81} . \mathbf{F}_5 is not generally available but we should be able to achieve $\sim 2^{64}$ cycles using FXL on a large SMP system.

We can easily see that we must be very careful choosing our parameters for security against one attack may expose it to another. *Our selected parameters are all tuned against this particular attack and this attack is no better or worse than direct attack, which have complexity of solving 24, and 28 equations in as many variables over GF(256), or roughly 2^{83} and 2^{98} respectively.*

But this is just a *unbalanced oil and vinegar attack*. The more efficiently implemented systems are Rainbow and have multiple layers. If we look at the Rainbow construction, it looks more like

$$M_i := \left[\begin{array}{ccc|ccc} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array} \right], \text{ if } i \leq m-o; \quad \left[\begin{array}{ccc|ccc} \alpha_{11}^{(i)} & \cdots & \alpha_{1v}^{(i)} & \alpha_{1,v+1}^{(i)} & \cdots & \alpha_{1n}^{(i)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{v1}^{(i)} & \cdots & \alpha_{vv}^{(i)} & \alpha_{v,v+1}^{(i)} & \cdots & \alpha_{vn}^{(i)} \\ \hline \alpha_{v+1,1}^{(i)} & \cdots & \alpha_{v+1,v}^{(i)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1}^{(i)} & \cdots & \alpha_{nv}^{(i)} & 0 & \cdots & 0 \end{array} \right], \text{ otherwise.} \tag{5}$$

That is, only the last o equations looks like Eq. [3], the initial $m - o$ equations actually have non-zero entries in the upperleft submatrix — which actually looks like a UOV matrix itself, i.e., has a block of zeros on the lower right. We don't bother with that detail. Can we exploit this property? Yes we can.

At this point, we should no longer consider T as the identity. Let us think about what the matrix M_T does in Rainbow. At the moment that we distill the P_n portion out, $m - o$ of the new M_i 's should show a zero last column. *However we don't*; M_T mixes the M_i 's together so that they in fact don't — we will see most of the time only the lower right entry as zero. But if we take any $o + 1$ of those last columns, there will be a non-trivial linear dependency. We can verify

that by setting one of those columns as the linear combination as the other o, the resulting equations are still quadratic!

Algorithm 3 (Rainbow Band Separation). *Reconciliation may be extended for a Rainbow scheme where the final stage has o oil and $v = n - o$ vinegar variables (which has the smaller indices):*

1. Perform basis change $w_i := w'_i - \lambda_i w'_n$ for $i = 1 \cdots v$, $w_i = w'_i$ for $i = v + 1 \cdots n$. Evaluate \mathbf{z} in \mathbf{w}' .
2. Find m equations by setting all coefficients of $(w'_n)^2$ to be zero; there are v variables in the λ_i 's.
3. Set all cross-terms involving w'_n in $\mathbf{z}_1 - \sigma_1^{(1)} \mathbf{z}_{v+1} - \sigma_2^{(1)} \mathbf{z}_{v+2} - \cdots - \sigma_o^{(1)} \mathbf{z}_m$ to be zero and find $n - 1$ more equations. Note that $(w'_n)^2$ terms are assumed gone already, so we can no longer get a useful equation.
4. Solve $m + n - 1$ quadratic equations in $o + v = n$ unknowns. We may use any method (e.g., \mathbf{F}_4 or XL).
5. Repeat the process to find P_{n-1} . Now set $w'_i := w''_i - \lambda_i w''_{n-1}$ for $i = 1 \cdots v$, and set every $(w''_{n-1})^2$ and $w''_n w''_{n-1}$ term to zero after making the substitution. Also set $\mathbf{z}_2 - \sigma_1^{(2)} \mathbf{z}_{v+1} - \sigma_2^{(2)} \mathbf{z}_{v+2} - \cdots - \sigma_o^{(2)} \mathbf{z}_m$ to have a zero second-to-last column. This time there are $2m + n - 2$ equations in n unknowns.
6. Continue in the same vein to find P_{n-2}, \dots, P_{v+1} .

The idea was mentioned by Mr. Yu-Hua Hu to one of the authors in a conversation, for which we are indebted. And this attack explains why the current parameter set suggested looks like that in Sec. 5.

Example 8. We run the attack on an instance of enTTS(16, 22) [25] which has the shape (6, 7, 1, 1, 7). The algebraic portion of the attack results in a system with 22 variables and 37 equations. This with XL at degree $D_{XL} = 6$ can be solved using 400MB (actually 415,919,856 bytes) of memory and 123,257 seconds on a 16-core, 2.2GHz Opteron machine with a total of 1,877,572 seconds of K8-CPU time. The number of multiplications is about 2^{47} , or ~ 16 cycles a multiplication.

On a single core, a K8 machine running XL-Wiedemann can average one multiplication in $\text{GF}(2^8)$ in about 9 cycles. The slowdown comes from the communications requirement between cores.

Example 9. The attack on an instance of enTTS(20, 28) [25] should result in a system with 22 variables and 37 equations. This with XL at degree $D_{XL} = 7$ should be solvable in 15GB of main memory and about 2^{56} multiplications. This is under the design complexity of 2^{72} .

We are also testing the prowess of other system-solving methods like Magma's \mathbf{F}_4 .

5.2 Interlinked/Accumulating Kernels and MinRank

As noted in [25] and recapped in Sec. 3, if μ combinations of central equations stays at the minrank, a Rank attack often speed up μ -fold, and which is termed *interlinking* or *accumulation* of kernels.

Recently Billet and Gilbert [4] cryptanalyzed the Rainbow instance of [7] in $\sim 2^{64}$ 3DES unit times (they stated 2^{71} , but GF(256)-multiplications is a very small unit; NESSIE for example counted 3DES units) using the same principle. While we exhibit a faster attack on that rainbow instance above, the same extended accumulating-kernel minrank attack is more widely applicable:

Proposition 3 (Billet-Gilbert). *Kernels of the initial block of equations in a rainbow-like multivariate always accumulate such that any vector in \mathbf{x} -space with the initial vinegar components all vanishing has at least a $1/q$ probability of being found by the MinRank attack.*

Example 10. We can cryptanalyze enTTS(20,28) [25] in 2^{64} via the accumulating kernels attack.

In fact, this pitfall is sometimes easy to overlook:

Proposition 4. *We can cryptanalyze TRMS from [20] in $\sim 2^{62}$ via the accumulating kernels attack.*

Proof. The central map has this piece with $*_3$ meaning multiplication in GF(2^{24}):

$$\begin{pmatrix} y_{17} \\ y_{18} \\ y_{19} \end{pmatrix} = \begin{pmatrix} x_{17} \\ x_{18} \\ x_{19} \end{pmatrix} *_3 \begin{pmatrix} x_8 \\ x_9 + x_{11} + x_{12} \\ x_{13} + x_{15} + x_{16} \end{pmatrix} + \begin{pmatrix} c_{29}x_4x_{16} \\ c_{30}x_5x_{10} \\ c_{31}x_{15}x_{16} \end{pmatrix} + \begin{pmatrix} c_{32}x_9 \\ c_{33}x_{10} \\ c_{34}x_{11} \end{pmatrix}.$$

Each of these equations are only of rank 8 (the minrank) in GF(256), and the y_{17} and y_{19} form a pair of equations that has $q = 256$ interlinked kernels. Evaluating as in Sec. 3 gives $\approx 2^{62}$.

In our schemes, the attack has complexity roughly q to the number of equations in the first block times change, which comes out to about $2^{85}, 2^{100}, 2^{118}$.

5.3 The Challenge

From all the above, we can see that we need to be very careful in our design of the parameter for Rainbow like schemes.

Proposition 5. *To build a scheme with design security C over the base field GF(q), we let ℓ be the smallest integer such that $q^{\ell+1} \gtrsim C$, then:*

- *The initial segment must contain $\ell - 1$ or more vinegar variables. The final segment must contain $\ell - 1$ or more equations and exactly as many as there are total vinegar variables.*
- *There should be enough equations to avoid direct solution via a Lazard-Faugère solver.*

Current estimate [24] is that 20 underdetermined equations in GF(2^8) achieves 2^{72} ; 24 equations achieves 2^{82} ; each extra equation roughly gives a factor $\gtrsim 2^{2.5}$ to the complexity [24].

We conclude that all three Rainbow like schemes we propose below have security levels above 2^{80} elementary operations. The best attack is with Algorithm 3, and the expected complexity in $\text{GF}(2^8)$ multiplications is 2^{84} , 2^{87} , 2^{80} respectively.

1. Rainbow (20,10,4,10), in the extended form $((15, 10), (4, 4), (1, 10))$
2. Rainbow (18,12,12), in the extended form $((17, 12), (1, 12))$
3. Rainbow (20,14,14), in the extended form $((19, 14), (1, 14))$.

Of course, without using the extended form, the security level would not be any lower, the extended form merely guarantees the existence of a signature always.

We hasten to add that the form given above is not much slower in signing than the previous TTS. In preliminary runs, a single signature for (20,10,4,10) version averages to about $157\mu\text{s}$, still way faster than any competitor.

6 Conclusion

In this paper, we present a new differential attack and a new Rainbow constructions. We design new schemes for practical applications.

With these constructions, we note that the design security of the system would still go up exponentially as the length of the hash in both generic (rainbow) and sparse (TTS) variants. *Perhaps, we might even say that the kinks of this approach is being ironed out, and multivariate cryptographers are finally beginning to understand Rainbow-like Multivariate Signatures.*

Another development that affects Rainbow-like schemes is the fact that SHA-1 is being phased out in the wake of recent results [21]. This means that hashes and hence signatures might become longer in a hurry. ECC is affected in much the same way, because 163- or 191-bit ECC may be obsoleted when everyone switches to SHA-2 (no one really wants to use a truncated hash if it can be helped). Even such state-of-the-art work as [3] would force the slightly uncomfortable SHA-224. With multivariate signature schemes, an additional problem is the large (and sometimes redundant, cf. [23]) keys. One might look toward other base fields such as $\text{GF}(16)$ to help with the key size problem, but this would also pose new challenges in optimization. Another way is to look for a safe TTS (built on the similar layer structures as specified above), now that hash sizes has gotten longer. Though the new attacks are found on Rainbow schemes, these attacks can be easily prevented by adjusting the parameter. All in all, we think that multivariates including Rainbow-like schemes still deserve a good look as the age of quantum computers approaches.

Acknowledgements

JD and BY are grateful to the Humboldt and Taft Foundations, and the Taiwan Information Security Center [National Science Council Project NSC 96-2219-E-011-008 / NSC 96-2219-E-001-001] without whose valuable support much of this work would not have been possible. BY would also like to thank NSC for partial sponsorship on Project NSC 96-2623-7-001-005-D.

Comments and correspondence: Please address to BY at by@moscito.org.

References

1. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving, pp. 71–74 (2004); Previously INRIA report RR-5049
2. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic expansion of the degree of regularity for semi-regular systems of equations. In: Gianni, P. (ed.) MEGA 2005, Sardinia (Italy) (2005)
3. Bernstein, D.J.: Curve25519: New diffie-hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
4. Billet, O., Gilbert, H.: Cryptanalysis of rainbow. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 336–347. Springer, Heidelberg (2006)
5. Coppersmith, D., Stern, J., Vaudenay, S.: The security of the birational permutation signature schemes. *Journal of Cryptology* 10, 207–221 (1997)
6. Courtois, N.T., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000); Extended Version: <http://www.minrank.org/xlfull.pdf>
7. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
8. Dubois, V., Fouque, P.-A., Shamir, A., Stern, J.: Practical cryptanalysis of sflash. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 1–12. Springer, Heidelberg (2007)
9. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra* 139, 61–88 (1999)
10. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: International Symposium on Symbolic and Algebraic Computation — ISSAC 2002, pp. 75–83. ACM Press, New York (2002)
11. Faugère, J.-C., Perret, L.: Polynomial equivalence problems: Algorithmic and theoretical aspects. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 30–47. Springer, Heidelberg (2006)
12. Garey, M.R., Johnson, D.S.: *Computers and Intractability — A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company (1979) ISBN 0-7167-1044-7 or 0-7167-1045-5
13. Geiselmann, W., Steinwandt, R., Beth, T.: Attacking the affine parts of SFlash. In: Honary, B. (ed.) *Cryptography and Coding 2001*. LNCS, vol. 2260, pp. 355–359. Springer, Heidelberg (2001); Extended version <http://eprint.iacr.org/2003/220/>
14. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
15. Joux, A., Kunz-Jacques, S., Muller, F., Ricordel, P.-M.: Cryptanalysis of the tractable rational map cryptosystem. In: PKC [19], pp. 258–274.

16. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
17. Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
18. Patarin, J., Goubin, L.: Trapdoor one-way permutations and multivariate polynomials. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 356–368. Springer, Heidelberg (1997); Extended Version <http://citeseer.nj.nec.com/patarin97trapdoor.html>
19. Vaudenay, S. (ed.): PKC 2005. LNCS, vol. 3386. Springer, Heidelberg (2005)
20. L.-C. Wang, Y.-H. Hu, F. Lai, C.y. Chou, and B.-Y. Yang. Tractable rational map signature. In PKC [19], pp. 244–257. ISBN 3-540-24454-9
21. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full sha-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
22. Wolf, C., Braeken, A., Preneel, B.: Efficient cryptanalysis of RSE(2)PKC and RSSE(2)PKC. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 294–309. Springer, Heidelberg (2005); Extended version <http://eprint.iacr.org/2004/237>
23. Wolf, C., Preneel, B.: Superfluous keys in Multivariate Quadratic asymmetric systems. In: PKC [19], pp. 275–287; Extended version <http://eprint.iacr.org/2004/361/>
24. Yang, B.-Y., Chen, J.-M.: All in the XL family: Theory and practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
25. Yang, B.-Y., Chen, J.-M.: Building secure tame-like multivariate public-key cryptosystems: The new TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)
26. Yang, B.-Y., Chen, O.C.-H., Bernstein, D.J., Chen, J.-M.: Analysis of QUAD. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 290–307. Springer, Heidelberg (2007)

Trapdoor Sanitizable Signatures and Their Application to Content Protection^{*}

Sébastien Canard¹, Fabien Laguillaumie², and Michel Milhau¹

¹ Orange Labs R&D, 42 rue des Coutures, BP6243, F-14066 Caen Cedex, France

² GREYC Université de Caen, Campus 2, Boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex, France

Abstract. Sanitizable signatures allow a designated entity to modify some specific parts of a signed message and to produce a new signature of the resulting message without any interaction with the original signer. In this paper, we extend these sanitizable signatures to formally introduce *trapdoor sanitizable signatures*. In this concept, the power of sanitization is given to possibly several entities, for a given message/signature by using a trapdoor computed by the signer at any time. We also give a generic construction of such trapdoor sanitizable signatures. Eventually, we apply our new cryptographic tool to group content protection, permitting members of the group to distribute a protected content among themselves.

1 Introduction

Digital Rights Management (DRM) systems provide efficient mechanisms to protect digital contents against unauthorized usages. Despite its increasing usage, in particular in video and music on-line services, users have difficulties to agree with device limitations induced by these systems. Consumers want flexibility with *legal* transfer or exchange of their acquired digital contents.

Let us consider the following scenario: a modern family made up of two parents, teenagers and children connected to a physical or wireless LAN. This family wants to buy some music songs offered by a well-known music service portal. The father identifies itself and purchases the desired protected media content on the portal of the license server. His local DRM agent installs in his device a license associated to this musical content. A license is a data structure which contains information about the authorized rights on the song and the cryptographic key necessary for decrypting the content. For security reasons this license has been strongly tied to the device. A license is protected by cryptographic functions: a part of it is encrypted with the public key of the recipient's device and consequently, it is the only device able to decrypt information helpful for a content use. In this situation, other family members could play this song only on the father's device.

^{*} This work has been financially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

This is a main drawback in spite of a few advances made in this field. Some private DRM applications allow the transfer of limited rights up to non connected devices like Personal Digital Assistant (i.e. Microsoft WMDRM). The normalization group OMA-DRM¹ itself defined the notion of domain to resolve exchange of rights/licenses inside a set of mobile devices belonging to a same domain. In fact each domain's device receives the same cryptographic key allowing the use of the domain license. The management of these domains and the generation of the associated keys are realized by the license server. This means that this server gets a specific knowledge about domain's members.

Similarly, iTunes from Apple enables buyers to transfer content to identified machines that share the same license decryption key. The decryption key is sent to a new computer from Apple's servers that can, by this way, limit the number of authorized computers (today limited to five).

To by-pass these restrictions, we propose a system which answers the following needs: (1) dynamicity of license transfers; (2) local management of the group members; (3) compliance with OMA-DRM architecture and protocols. The first and second requirements are incompatible with OMA-DRM domains; in fact we need to modify the former license received by the family's father. The DRM agent of the father will be able to create a new license targeted at another family member. This modification of license requires both operations: (1) decrypt the Content Encryption Key *CEK* thanks to the father's private key, and encrypt it again with the public key of the new recipient, (2) sign the new license such that the DRM agent of the recipient recognizes this signature as a true license server signature.

Technically, to get rid of this drawback, we need to adapt and develop a new type of signature scheme where it is possible for a designated user to modify some part of a message signed by a particular signer. The signature on the new message is still seen as a message signed by the initial signer. In DRM systems, the signer of a license is the license server and applying the new signature scheme, the result is that the OMA-DRM agent of the final user will accept the signature of the derived license as a real license server signature. Moreover, it should be possible to give this particular power later (*i.e.* not at the creation of the license) to permit users to change their mind when they want to.

1.1 Related Work

A usual way to expose declassified documents and keep a protection of intelligence sources is to blot out the sensitive parts. Well-known examples were the memo to US President that had been declassified for an inquiry into the 11 September 2001 terrorist attack and a US Department of Defense memo about who helped Iraq to militarize civilian Hugues helicopters. These examples of sanitized documents became famous because Naccache and Whelan demasked the blotted out words [13].

¹ Open Mobile Alliance is a standard consortium which develops open standards for the mobile phone industry.

In this example, keeping secret the sanitized part is fundamental. When applied to electronic documents, sanitizing can of course easily resist Naccache and Whelan attack, but an important issue is the *authentication* and *integrity* of such sanitized documents. If the original document is signed with a traditional signature scheme, modifying this document in any way will make the original signature be invalid. If the authentication must be preserved, a traditional signature cannot be used in this case. Of course, the signer could sign the sanitized document, but when applied to declassified documents for instance, the signer's secret key might have expired, or the signer may not be available at all. *Sanitizable signatures* have been introduced to address this problem.

Two different flavors of sanitizable signatures can be found in the literature. The first kind illustrates the previous scenario, namely the signature of the original message can be modified, without the help of the signer, to be also valid for the sanitized document [16,7,15]. In other words, a designated user can erase some part of a signed message and produce a new signature in such a way that the resulting signature is seen as a correct signature on the new message from the initial signer.

The other kind of sanitizable signatures was introduced in [1] by Ateniese, Chou, de Medeiros and Tsudik, also in terms of sanitizable signatures. Contrary to the above case, such signatures allow a semi-trusted censor to *modify* (not only to erase) some specific portions of a signed message and to produce a new valid signature of the resulting message without any interaction with the original signer. According to Ateniese *et al.*, a sanitizable signature scheme must ensure (1) *immutability*, which means that the censor must not be able to modify any part of the message, not specified by the signer, (2) *privacy*, which means that all sanitized information is unrecoverable, (3) *accountability*, which means that in case of dispute, the signer can prove to the court that a given message was sanitized, and (4) *transparency* which means that no one, except the signer and the censor, can guess whether a message has been sanitized. As we will see in section 3.1, Ateniese *et al.*'s scheme can easily be obtained *via* a sanitizable signature scheme of the first kind.

In [9], Klonowski and Lauks propose some improvement of the Ateniese *et al.* scheme [1] by proposing (in particular) generic techniques permitting first to limit the set of possible modifications of a single mutable block using either accumulator schemes or bloom filters and second to limit the number of modifications of mutable blocks. Note that their methods cannot be applied to our scheme since they do not use a trapdoor.

1.2 Our Contribution and Organization of the Paper

The sanitizable signature of Ateniese *et al.* may be useful for our application of content protection. But in fact, the proposed properties of their sanitizable signature scheme are not sufficient. More precisely, we need the possibility for any authorized user to have a special trapdoor to modify some designated parts of a signed message. Therefore, contrary to the modelization of Ateniese *et al.*, the power of the sanitizer is not given during the signature process.

In this paper, we thus define in Section 2 the security model of a new type of sanitizable signatures, called *trapdoor* sanitizable signatures, where the power of sanitization is given to possibly several entities, for a given message/signature by using a trapdoor issued by the signer at any time.

We also give the first generic construction in Section 3 of such sanitizable signature scheme based on an identity-based chameleon hashing function and on a classical signature scheme. We then give a possible instance of our generic construction and apply in Section 4 our new cryptographic tool to content protection so as to fit our requirement for a DRM license, as described previously.

2 Trapdoor Sanitizable Signature Scheme

In a trapdoor sanitizable signature scheme, the signer allows a specific user to modify a portion of a signed message by producing a piece of information that will help this user in sanitizing the document. This trapdoor information is given upon the will of the signer, who can choose *to whom* and *when* he will deliver it. This last property makes a crucial difference with conventional sanitizable signatures, and is of importance to design our DRM scheme.

In this section, we first briefly and informally describe sanitizable signatures as they appear in the literature, and then, we propose a formal definition and a precise security model for our new primitive of *trapdoor* sanitizable signatures.

2.1 Review of Existing Definitions for Sanitizable Signatures

We recall here several definitions for sanitizable signature schemes to enlight the different notions and the difference with our new concept.

- According to [16,7,11,12,15], a sanitizable signature scheme is a scheme which allows a specific user (not necessarily chosen by the signer) to sanitize certain portions of a message (which means that these portions become unavailable) and to generate a valid signature for this new document, without any interaction with the signer. The scheme consists of four procedures: the key generation, the signing process, the sanitizing phase, and the verification. This scheme must resist an existential forgery under a chosen message attack, and must be indistinguishable under a chosen message attack (which means that an attacker is not able to distinguish between two signatures, which one is one a sanitized message).
- According to [19], a sanitizable signature scheme permits a specific user to modify a message (*a priori* chosen by the signer) and to produce a valid signature for the new message. The main difference with the previous definition is therefore the possibility to replace some parts of the documents with some others. The different algorithms which constitute the scheme are the same as in the previous definition. The scheme must be existentially unforgeable under a chosen message attack, and the authors also propose a notion of indistinguishability and one of *identical distribution* between the distribution coming from the signing algorithm and the one coming from the sanitizing algorithm.

2.2 Definition of a Trapdoor Sanitizable Signature Scheme

We present in this section a formal definition for *trapdoor sanitizable signatures* and its security model. The main difference with Ateniese *et al.*'s definition is that the original signer produces a trapdoor information, depending on a specific message divided into several blocks of independent sizes, that he will transmit to the sanitizer, who will be able to *modify* this message.

Definition 1 (Trapdoor sanitizable signature scheme). A trapdoor sanitizable signature scheme TSS consists in the following algorithms.

- *Setup* is a probabilistic algorithm which takes a security parameter k as input and outputs the public parameters \mathcal{P} . $\mathcal{P} \leftarrow \text{Setup}(k)$.
- *KeyGen* is a probabilistic algorithm which takes the public parameters \mathcal{P} as input and outputs a pair of secret and public keys (sk, pk) . $(sk, pk) \leftarrow \text{KeyGen}(\mathcal{P})$.
- *Sign* is a probabilistic algorithm which takes public parameters \mathcal{P} , a message $m = m_1 \parallel \dots \parallel m_L$ and a secret key sk as inputs, and outputs a signature σ on the message m and the set $I \subset \llbracket 1, L \rrbracket$ of the indices that are sanitizable on this signature. $(\sigma, I) \leftarrow \text{Sign}(\mathcal{P}, m, sk)$.
- *Trapdoor* is a deterministic algorithm which takes public parameters \mathcal{P} , a message m and a valid signature σ and a secret key sk as inputs and outputs a trapdoor \mathbf{t} . $\mathbf{t} \leftarrow \text{Trapdoor}(\mathcal{P}, m, \sigma, sk)$.
- *Sanitize* is an algorithm which takes public parameters \mathcal{P} , a message m , a valid signature σ on m under the public key pk , a message \tilde{m} , the set I of the indices that are sanitizable and a trapdoor \mathbf{t} and outputs a signature $\tilde{\sigma}$ on the message \tilde{m} . $\tilde{\sigma} \leftarrow \text{Sanitize}(\mathcal{P}, m, \sigma, pk, \tilde{m}, I, \mathbf{t})$.
- *Verif* is a deterministic algorithm which takes public parameters \mathcal{P} , a message m , a putative signature σ , a public key pk and the set I of the indices that are sanitizable as inputs and outputs 1 if the signature σ on m is valid and 0 otherwise. $0/1 \leftarrow \text{Verif}(\mathcal{P}, m, \sigma, pk, I)$.

Remark 1. Note that the *Sanitize* algorithm can be either deterministic or probabilistic. In fact, in all existing constructions (including ours), as the construction is based on the use of chameleon hash functions, this algorithm is deterministic. But we can easily imagine that it is possible to design (trapdoor) sanitizable signature schemes with a probabilistic *Sanitize* algorithm.

The security criteria which must be fulfilled are discussed in the following paragraph.

2.3 Security Model

Correctness. A trapdoor sanitizable signature scheme must satisfy two *correctness* properties:

$$\forall k \in \mathbb{N}, \forall \mathcal{P} \leftarrow \text{TSS.Setup}(k), \forall (pk, sk) \leftarrow \text{TSS.KeyGen}(\mathcal{P}), \\ \forall (\sigma, I) \leftarrow \text{TSS.Sign}(\mathcal{P}, m, sk), \text{Verif}(\mathcal{P}, m, \sigma, pk, I) \rightarrow 1$$

and

$$\forall k \in \mathbb{N}, \forall \mathcal{P} \leftarrow \text{TSS.Setup}(k), \forall (pk, sk) \leftarrow \text{TSS.KeyGen}(\mathcal{P}), \\ \forall (\sigma, I) \leftarrow \text{TSS.Sign}(\mathcal{P}, m, sk), \forall \mathbf{t} \leftarrow \text{Trapdoor}(\mathcal{P}, m, \sigma, sk), \\ \forall \tilde{\sigma} \leftarrow \text{Sanitize}(\mathcal{P}, m, \sigma, pk, \tilde{m}, I, \mathbf{t}), \text{Verif}(\mathcal{P}, \tilde{m}, \tilde{\sigma}, pk, I) \rightarrow 1$$

Unforgeability. A trapdoor sanitizable signature scheme must also satisfy an unforgeability property. The conventional notion of security for signatures was introduced by Goldwasser, Micali and Rivest in [6]. A signature scheme must be *existentially unforgeable* under a *chosen message attack*. We present here the formal definition of existential unforgeability under a chosen message attack (EU-CMA) for trapdoor sanitizable signatures.

First, we need to define the following oracles.

- $\mathcal{O}^{\text{Sign}}$: this oracle is initialized with a public key pk and the corresponding secret signing key sk . It takes as input a message m and outputs a valid signature related to this message and the public key pk .
- $\mathcal{O}^{\text{Trapdoor}}$: this oracle is initialized with a public key pk and the corresponding secret signing key sk . It takes as input a message m and a signature σ . If $\text{Verif}(\mathcal{P}, m, \sigma, pk, I) = 0$, then the oracle outputs **error**. Otherwise, it outputs a trapdoor \mathbf{t} related to the message m , the signature σ and the public key pk as if it is output by the **Trapdoor** algorithm.
- $\mathcal{O}^{\text{Sanitize}}$: this oracle is initialized with a public key pk and the corresponding secret signing key sk . It takes as input two messages m and \tilde{m} and a signature σ . If $\text{Verif}(\mathcal{P}, m, \sigma, pk, I) = 0$, then the oracle outputs **error**. Otherwise, it computes a trapdoor \mathbf{t} related to the message m , the signature σ and the public key pk (using sk) and outputs the signature $\tilde{\sigma}$ on \tilde{m} as if it is output by the **Sanitize** algorithm on input m , σ and \tilde{m} . Note that the trapdoor \mathbf{t} can be either deleted or not at the end of the request.

We say that a trapdoor sanitizable signature scheme is *existentially unforgeable under a chosen message attack*, if no PPT adversary \mathcal{F} has a non negligible success in the following game:

1. The challenger \mathcal{C} runs the **Setup** algorithm to produce the public parameters and then runs the **KeyGen** algorithm. It obtains the pair of keys (pk^*, sk^*) to be attacked and gives the public key pk^* to \mathcal{F} .
2. The forger \mathcal{F} adaptively interacts with the signing oracle $\mathcal{O}^{\text{Sign}}$ and the trapdoor oracle $\mathcal{O}^{\text{Trapdoor}}$.

3. Eventually, \mathcal{F} comes up with a message m^* and a signature σ^* . \mathcal{F} is said to *succeed* if the pair (m^*, σ^*) verifies the four following properties.
 - (a) $\text{Verif}(\mathcal{P}, m^*, \sigma^*, pk^*, I) \rightarrow 1$.
 - (b) (m^*, σ^*) does not come from the $\mathcal{O}^{\text{Sign}}$ oracle.
 - (c) (m^*, σ^*) does not come from the $\mathcal{O}^{\text{Sanitize}}$ oracle.
 - (d) (m^*, σ^*) is not linked to a tuple (\mathbf{t}, m, σ) from the $\mathcal{O}^{\text{Trapdoor}}$ oracle. More precisely, for all message m being in input of the $\mathcal{O}^{\text{Trapdoor}}$ oracle, we should have that $\exists i \notin I_m, m_i^* \neq m_i$ where I_m corresponds to the set I output with the signature σ and related to m .

Remark 2. Note that by describing condition (3d) like that, we reject the possibility for the adversary to forge a signature on a message related to one for which a trapdoor has been asked. This can be seen as restrictive (we don't know if the adversary has used the trapdoor or not to produce the forge) but this is not really a problem in practice. Note also that if the Sanitize algorithm is deterministic (see above), this is easily verifiable since the corresponding signature can be also computed by the challenger.

The *success* of \mathcal{F} is defined as the probability (over all internal random coins) of its success in this previous game. \mathcal{F} is said to $(q_S, q_T, q_{S_z}, \tau, \varepsilon)$ -breaks the existential unforgeability in the chosen message attack of the trapdoor sanitizable signature scheme, if its success is ε , its running time is τ , and its number of queries to the signing oracle (resp. trapdoor oracle and sanitize oracle) is q_H (resp. q_T and q_{S_z}).

Indistinguishability. We require that values produced by the Sanitize algorithm are distributed identically to those produced by the Sign algorithm. In particular, the following distributions $\mathcal{D}_{\text{Sanit}}$ and $\mathcal{D}_{\text{Sign}}$ are indistinguishable for all \mathcal{P}, pk, sk :

$$\mathcal{D}_{\text{Sanit}} = \{ \tilde{\sigma} : \tilde{\sigma} = \text{Sanitize}(\mathcal{P}, m, \sigma, pk, \tilde{m}, I, \mathbf{t}), (m, \tilde{m}) \in \mathcal{M}, (\sigma, I) = \text{Sign}(\mathcal{P}, m, sk), \mathbf{t} = \text{Trapdoor}(\mathcal{P}, m, \sigma, sk) \}$$

and

$$\mathcal{D}_{\text{Sign}} = \{ \sigma : (\sigma, I) = \text{Sign}(\mathcal{P}, m, sk), m \in \mathcal{M} \}.$$

3 Generic Construction

Before describing a generic construction of a trapdoor sanitizable signature scheme, we will first give an intuition of this construction by revisiting a bit Ateniese *et al.*'s scheme from [1] in the light of a sanitizable scheme of Miyazaki *et al.* [11,12]. As we already explained, the corresponding two definitions of sanitizable signatures are different. Nevertheless, the next section shows how Ateniese *et al.*'s scheme can be obtained from Miyazaki *et al.*'s one.

3.1 Ateniese *et al.*'s Sanitizable Signatures Revisited

Let's first recall the scheme SUMI-4 from Miyazaki *et al.*'s [11,12]. The rough idea is to replace the parts of the message that are censored by hash values of these parts. Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verif})$ be a signature scheme and h be a hash function. SUMI-4 works as follows.

Signer

1. Let $m = m_1 \parallel \dots \parallel m_L$ be the message to sign, the signer first picks random values $r_i \in \{0, 1\}^k$ for $i = 1 \dots L$.
2. The signer generates a set $H = \{h_i\}_{i=1 \dots L}$ where $h_i = h(m_i, r_i)$
3. The signature is obtain as $\sigma = \text{Sign}(H)$.
4. The signer outputs $(\{m_i, r_i\}_{i=1 \dots L}, \sigma)$

Sanitizer

1. Determine $I \subset [1, L]$ the indices of the message to be censored.
2. The sanitizer converts the document m to $\tilde{m} = \tilde{m}_1 \parallel \dots \parallel \tilde{m}_L$ where

$$\tilde{m}_i = \begin{cases} (m_i, r_i) & \text{if } i \notin I \\ h_i & \text{if } i \in I \end{cases}$$

3. The sanitizer outputs (\tilde{m}, I)

Verifier

1. Let (\tilde{m}, I) the sanitized message and σ the original signature. The verifier generates the set of hash values $\tilde{H} = \{\tilde{h}_i\}_{i=1 \dots L}$ such that

$$\tilde{h}_i = \begin{cases} h(\tilde{m}_i) & \text{if } i \notin I \\ \tilde{m}_i & \text{if } i \in I \end{cases}$$

2. The verifier then checks $\text{Verif}(\tilde{H}, \sigma)$ to verify the validity of the signature.

Now, let's look at this scheme from Ateniese *et al.*'s point of view. In their definition of sanitizable signatures, a specific user can not only erase a part of the message, but he can replace it by something else. Suppose that h is replaced by a *chameleon hash function*, as introduced by Krawczyk and Rabin in [10]. This means that a pair of secret and public key parametrizes this hash function, with the property that the owner of the secret key is able to find a collision to a hash value computed thanks to his public key. In this setting, the signer *chooses* the sanitizer (by selecting his "chameleon" public key) and replaces the function h by the chameleon hash function h^C . The sanitizer is then able to replace the sanitized parts by values of his choice, thanks to his secret key. Indeed, given a message m'_i (for $i \in I$), he can compute an element r'_i such that

$$h^C(m'_i, r'_i) = h^C(m_i, r_i) = h_i.$$

The remaining works as follows: the sanitizer converts the document m to $\tilde{m} = \tilde{m}_1 \| \dots \| \tilde{m}_L$ where

$$\tilde{m}_i = \begin{cases} (m_i, r_i) & \text{if } i \notin I \\ (m'_i, r'_i) & \text{if } i \in I \end{cases}$$

and the verifier generates the set of hash values $\{\tilde{h}_i\}_{i=1\dots L}$ such that $\tilde{h}_i = h^C(\tilde{m})$ and eventually checks $\text{Verif}(H', \sigma)$ to verify the validity of the signature.

This is another look at Ateniese *et al.*'s scheme. Our idea to design trapdoor sanitizable signature is to replace the chameleon hash function by an *identity-based* chameleon hash function. The secret key extraction from an identity will actually help us to generate the trapdoor allowing the sanitization of a specific message. The following section rigorously describes our construction relying on this idea.

3.2 A Generic Construction of *Trapdoor* Sanitizable Signatures

We propose in this section a generic construction which uses as building blocks an identity-based chameleon hash function [10] to achieve our “trapdoor” requirement. Roughly speaking a chameleon hash function is a trapdoor hash function, such that the owner of the trapdoor is able to find collisions for every given input. When they are used instead of traditional hash functions in a signature scheme based on the well-known *hash-and-sign* paradigm, the resulting scheme is a *chameleon signature scheme* which is highly related to Chaum and van Antwerpen’s *undeniable signatures* [5] and Jakobsson, Impagliazzo and Sako’s *designated verifier signatures* [8].

Identity-Based Chameleon Hashing. Identity-based chameleon hashing were introduced by Ateniese and de Medeiros [2]. We assume that it is possible to identify all systems to a bit-string easily derivable from a system’s public knowledge. We call such a string an identity string and we note it Id . There are two actors in such schemes: an authority \mathcal{A} and a user \mathcal{U} . For mally, an *identity-based chameleon hash scheme* CH is defined by the following family of efficiently computable algorithms.

- **Setup:** this probabilistic algorithm is executed by \mathcal{A} to generate a pair of key sk_{CH} and pk_{CH} , taking on input a security parameter k .

$$(sk_{CH}, pk_{CH}) \leftarrow \text{Setup}(k)$$
- **Extract:** this probabilistic algorithm, executed by \mathcal{A} taking on inputs the identity Id of a user \mathcal{U} and the secret key sk_{CH} , outputs the derived trapdoor information B .

$$B \leftarrow \text{Extract}(Id, sk_{CH})$$
- **Proceed:** this probabilistic algorithm that can be executed by anybody and which takes on inputs the public key pk_{CH} , the identity Id of a user, a message m and a random value r and outputs the hash value h .

$$h \leftarrow \text{Proceed}(pk_{CH}, Id, m, r)$$

- **Forge:** this algorithm is executed by the user with identity Id to compute a hash value on a new message. It takes as inputs the public key pk_{CH} , the identity Id , the corresponding extracted value B , a new message m' and the hash value h on a message m with random value r and it outputs a new hash value (h, r') .

$$r' \leftarrow \text{Forge}(pk_{CH}, Id, B, m', r, h, m)$$

This construction must be *correct*, which means that if $B = \text{Extract}(Id, sk_{CH})$ and if we assume that $h = \text{Proceed}(pk_{CH}, Id, m, r)$ where m is a message and r a random value, then $h = \text{Proceed}(pk_{CH}, Id, m', \text{Forge}(pk_{CH}, Id, B, m', r, h, m))$.

We need to introduce the following new security property for the identity-based hash function, to prove the security of our scheme. This notion is not defined in Krawczyk and Rabin’s paper [10] nor in the one of Ateniese et de Meideros [2], but it is a natural generalization of the notion of collision for traditional hash functions. An Id-based chameleon hash function must be *collision-resistant*, in the sense of the following game:

1. A challenger \mathcal{C} runs the **Setup** algorithm to produce the pair of keys (pk^*, sk^*) and gives the public key pk^* to \mathcal{F} .
2. The collision finder \mathcal{F} adaptively interacts with an **Extract** oracle $\mathcal{O}^{\text{Extract}}$ to obtain a trapdoor information B corresponding to some identity string.
3. Eventually, \mathcal{F} comes up with a tuple (m, m', r, r', Id) . \mathcal{F} is said to *succeed* if $\text{Proceed}(pk^*, Id, m, r) = \text{Proceed}(pk^*, Id, m', r')$ and Id has not been proposed to the **Extract** oracle.

Moreover, it is required that the distributions of r and r' must be the same (random and uniform) and that a message m induces the same probability distribution on $\text{Proceed}(pk^*, Id, m, r)$ for a random r (cf. Krawczyk and Rabin’s *uniformity* from [10]).

The Construction. Our construction also relies on a classical signature scheme $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verif})$ which follows the “hash-and-sign” paradigm. That is, from a key pair (sk_S, pk_S) , output by the $\Sigma.\text{KeyGen}$ algorithm, and a message m , the signature σ is computed as $\sigma = \Sigma.\text{Sign}(sk_S, m)$. Eventually, from a signature σ , a message m and a public key pk_S , the verifier checks that $\Sigma.\text{Verif}(pk_S, m, \sigma) = 1$. This signature scheme must be existentially unforgeable under a chosen message attack as defined in [6].

- **Setup:** the **Setup** consists in executing $\Sigma.\text{Setup}$ to output some public parameters.
- **KeyGen:** the **KeyGen** phase consists in executing the $\text{CH.Setup}(k)$ for the chameleon hash function and the $\Sigma.\text{KeyGen}(k)$ for the signature scheme. Consequently, the global secret key of the trapdoor sanitizable scheme is $sk = (sk_{CH}, sk_S)$ and the corresponding public key is $pk = (pk_{CH}, pk_S)$.

```

Procedure KeyGen( $k$ ):
    ( $sk_{CH}, pk_{CH}$ )  $\leftarrow$  CH.Setup( $k$ )
    ( $sk_S, pk_S$ )  $\leftarrow$   $\Sigma$ .KeyGen( $k$ )
     $sk = (sk_{CH}, sk_S)$ 
     $pk = (pk_{CH}, pk_S)$ 
    Output ( $sk, pk$ )
    
```

- **Sign**: the Sign step first consists in choosing the set I of the indices that are sanitizable. For each $i \in I$, we execute CH.Proceed(pk_{CH}, m, m_i, r_i) which outputs h_i . For all $i \in \llbracket 1, L \rrbracket$, we set $\hat{m}_i = m_i$ if $i \notin I$ and $\hat{m}_i = h_i$ otherwise. We denote by $\hat{m} = \hat{m}_1 \parallel \dots \parallel \hat{m}_L$. σ is the concatenation of the output s of the Σ .Sign algorithm on input sk_S and \hat{m} and of all elements of the set R . We also add a verification value h_c so as to prevent some types of attacks. The output of the algorithm is finally σ and the set I .

```

Procedure Sign( $\mathcal{P}, m, sk$ ):
     $m_1 \parallel \dots \parallel m_L \leftarrow m$ 
    Set  $I \subset \llbracket 1, L \rrbracket$ 
     $\forall i \in I, r_i \in_R \mathcal{R}$ 
     $\forall i \in I, h_i \leftarrow$  CH.Proceed( $pk_{CH}, m, m_i, r_i$ )
     $\forall i \in \llbracket 1, L \rrbracket \setminus I, \hat{m}_i \leftarrow m_i$ 
     $\forall i \in I, \hat{m}_i \leftarrow h_i$ 
     $r_c \in_R \mathcal{R}$ 
     $h_c \leftarrow$  CH.Proceed( $pk_{CH}, m, m, r_c$ )
     $\hat{m} = \hat{m}_1 \parallel \dots \parallel \hat{m}_L \parallel h_c$ 
     $R \leftarrow \{r_i : i \in I\}$ 
     $\sigma \leftarrow \Sigma$ .Sign( $sk_S, \hat{m}$ )
     $\forall i \in I, \sigma \leftarrow \sigma \parallel r_i$ 
     $\sigma \leftarrow \sigma \parallel r_c$ 
    Output ( $\sigma, I$ )
    
```

- **Trapdoor**: the Trapdoor function consists in executing the CH.Extract of the Id-based chameleon hash function with m as the identity and sk_{CH} . It outputs the derived trapdoor information \mathbf{t} .

```

Procedure Trapdoor( $\mathcal{P}, m, \sigma, sk$ ):
    if Verif( $\mathcal{P}, m, \sigma, pk, I$ ) = 1 then
         $\mathbf{t} \leftarrow$  CH.Extract( $m, sk_{CH}$ )
        Output  $\mathbf{t}$ 
    otherwise output error
    
```

- **Sanitize**: let $\tilde{m} = \tilde{m}_1 \parallel \dots \parallel \tilde{m}_L$. Remember that σ is the concatenation of s and all elements of $R = \{r_i : i \in I\}$. The CH.Forge algorithm is executed for all $i \in I$ on input pk_{CH}, m, \mathbf{t} , the new message \tilde{m}_i, r_i, h_i and m_i that

outputs each time a new \tilde{r}_i and we do the same for the verification value h_c . The new signature $\tilde{\sigma}$ is thus the concatenation of s (unchanged) and all elements of $\tilde{R} = \{\tilde{r}_i : i \in I\}$ plus r_c .

```

Procedure Sanitize( $\mathcal{P}, m, \sigma, pk, \tilde{m}, I, \mathbf{t}$ ):

 $m_1 \parallel \dots \parallel m_L \leftarrow m$ 
 $\tilde{m}_1 \parallel \dots \parallel \tilde{m}_L \leftarrow \tilde{m}$ 
Retrieve  $s, R$  and  $r_c$  from  $\sigma$ 
 $\forall i \in I, h_i \leftarrow \text{CH.Proceed}(pk_{CH}, m, m_i, r_i)$ 
 $\forall i \in I, \tilde{r}_i \leftarrow \text{CH.Forge}(pk_{CH}, m, \mathbf{t}, \tilde{m}_i, r_i, h_i, m_i)$ 
 $\tilde{R} \leftarrow \{\tilde{r}_i : i \in I\}$ 
 $h_c \leftarrow \text{CH.Proceed}(pk_{CH}, m, m, r_c)$ 
 $\tilde{r}_c \leftarrow \text{CH.Forge}(pk_{CH}, m, \mathbf{t}, \tilde{m}, r_c, h_c, m)$ 
 $\sigma \leftarrow s$ 
 $\forall i \in I, \sigma \leftarrow \sigma \parallel \tilde{r}_i$ 
 $\sigma \leftarrow \sigma \parallel \tilde{r}_c$ 
Output  $\sigma$ 
    
```

- **Verif**: the verification procedure consists in computing h_i for each m_i with $i \in I$ by using CH.Proceed on input $pk_{CH}, m = m_1 \parallel \dots \parallel m_L, m_i$ and r_i . For all $i \in \llbracket 1, L \rrbracket$, we set $\hat{m}_i = m_i$ if $i \notin I$ and $\hat{m}_i = h_i$ otherwise and we denote by $\hat{m} = \hat{m}_1 \parallel \dots \parallel \hat{m}_L \parallel h_c$. The output of the **Verif** algorithm is the output of $\Sigma.\text{Verif}(pk_S, s, \hat{m}) = 1$.

```

Procedure Verif( $\mathcal{P}, m, \sigma, pk, I$ ):

Retrieve  $s, R = \{r_i : i \in I\}$  and  $r_c$  from  $\sigma$ 
 $m_1 \parallel \dots \parallel m_L \leftarrow m$ 
 $\forall i \in I, h_i \leftarrow \text{CH.Proceed}(pk_{CH}, m, m_i, r_i)$ 
 $\forall i \in \llbracket 1, L \rrbracket \setminus I, \hat{m}_i \leftarrow m_i$ 
 $\forall i \in I, \hat{m}_i \leftarrow h_i$ 
 $h_c \leftarrow \text{CH.Proceed}(pk_{CH}, m, m, r_c)$ 
 $\hat{m} = \hat{m}_1 \parallel \dots \parallel \hat{m}_L \parallel h_c$ 
Output  $\Sigma.\text{Verif}(pk_S, s, \hat{m})$ 
    
```

3.3 Security

First of all, the correctness of our scheme is obvious. We will then concentrate our security analysis on unforgeability and indistinguishability.

Theorem 1 (Unforgeability)

Let \mathcal{F} be a $(q_S, q_T, q_{S_z}, \tau, \varepsilon)$ -forger against our trapdoor sanitizable signature scheme. Then there exists a $(\varepsilon', \tau', q_S)$ -existential forger \mathcal{F}' against the underlying signature scheme and a $(\varepsilon'', \tau'', q_T + q_{S_z})$ -collision finder \mathcal{C} against the identity-based hash function and a such that:

$$\varepsilon \leq \frac{1}{2}(\varepsilon' + \varepsilon'') \text{ and } \tau \geq \max\{\tau' + (q_T + q_{S_z})t_{\text{collision}}, \tau'' + q_S t_{\text{sign}}\}$$

where $t_{\text{collision}}$ and t_{sign} are the necessary time to compute a collision and to sign a message respectively.

Proof. To simplify the proof (and in particular to get rid of the set I of indices in the proof), we will suppose that a message consists of a single block². The condition (3d) of acceptance of the forge by the adversary needs then to be modified, taking into account that our Sanitize algorithm is deterministic. More precisely, considering the forge (m^*, σ^*) output by the adversary, we can easily test, for all message (m, σ) for which the adversary has asked the $\mathcal{O}^{\text{Trapdoor}}$ oracle, whether or not the Sanitize algorithm, on input $m, \sigma, m^*, \mathbf{t}$ output σ^* . If this is the case, the output of the adversary is rejected.

Let's suppose that there exists a $(q_S, q_T, q_{S_z}, \tau, \varepsilon)$ -forger against our trapdoor sanitizable signature scheme. Given a public key pk as input, after at most q_S queries to the signing oracle, q_T queries to the trapdoor oracle, and q_{S_z} queries to the sanitize oracle, the forger outputs a pair m^*, σ^* , with σ^* corresponding to a random coin r^* . For the forger to win the game, two possibilities arise:

- case 1: m^* is a non-sanitized message and σ^* has not been obtained from the signing oracle
- case 2: m^* is a sanitized message which has not been formed with a trapdoor information obtained from the trapdoor oracle nor the sanitize oracle. The pair (σ^*, r^*) must not come from the signing oracle. Nevertheless, we can suppose that σ^* comes from the signing oracle.

We will show that the first case allows the construction of a forger against the signature scheme Σ , the second allows the construction of a collision-finder on the identity-based hash function CH. We will exhibit a reduction \mathcal{R} which will flip a coin $b \in \{0, 1\}$ to bet which case will happen.

Case 1. Let's first consider the case 1 where the reduction designs an existential forger \mathcal{F}' . \mathcal{R} will use \mathcal{F} as a sub-routine to build this forger, which has as input a public key pk^* and some global parameters \mathcal{P} , and has to produce an existential forgery related to this public key. First, \mathcal{R} generates a pair of key (pk_{CH}, sk_{CH}) for the identity-based chameleon hash function thanks to CH.Setup. Then he sets $pk = (pk^*, pk_{CH})$ and gives this public key to \mathcal{F} . \mathcal{R} now has to simulate \mathcal{F} 's signing, trapdoor and sanitize oracles. To simulate the trapdoor and the sanitize oracles $\mathcal{O}^{\text{Trapdoor}}$ and $\mathcal{O}^{\text{Sanitize}}$, the reduction \mathcal{R} uses its knowledge of the trapdoor secret key sk_{CH} to create the trapdoor (and, in case of $\mathcal{O}^{\text{Sanitize}}$, computes the corresponding signature). To simulate the signing oracle $\mathcal{O}^{\text{Sign}}$, \mathcal{R} must answer to \mathcal{F} 's query related to a message m . It forwards this query to \mathcal{F} 's signing oracle which is parametrized by the secret key sk , and therefore

² However, this does not prevent an adversary to create new signatures without the knowledge of the trapdoor. The proof can be generalizable but will not be detailed in this paper due to space constraints.

\mathcal{R} simulates perfectly \mathcal{F} 's environment. At the end of the game, \mathcal{F} outputs a pair (m^*, σ^*) (with corresponding random bits r^*). \mathcal{R} then sets (m^*, σ^*) as \mathcal{F} 's outputs. This is clearly a successful forgery.

Case 2. Now, we consider that the reduction designs a collision-finder \mathcal{C} against the identity-based chameleon hash function CH from case 2. The reduction \mathcal{R} has to deal with \mathcal{C} 's challenge : namely a public key pk_{CH}^* for the chameleon hash function. \mathcal{R} then executes Σ .Setup and Σ .KeyGen to obtain the system's global parameters and a pair of keys (sk_Σ, pk_Σ) for the signature scheme Σ . As before, \mathcal{R} constructs a public key for the trapdoor sanitizable signature scheme $pk = (pk_{CH}^*, pk_\Sigma)$, which he forwards to the forger \mathcal{F} . To simulate the signing oracle, the reduction \mathcal{R} uses the secret key sk_Σ . When the forger \mathcal{F} queries a trapdoor for a pair (m, σ) , \mathcal{R} first checks the validity of the signature with the public key pk_Σ and then asks \mathcal{C} 's extractor oracle $\mathcal{O}^{\text{Trapdoor}}$ with m as the (identity) request. $\mathcal{O}^{\text{Trapdoor}}$'s answer is a trapdoor \mathbf{t} which corresponds to a correct trapdoor for sanitization. When the forger \mathcal{F} queries a sanitization for a message m and its signature σ , \mathcal{R} first checks the validity of the signature with the public key pk_Σ , secondly asks \mathcal{C} 's extractor oracle $\mathcal{O}^{\text{Trapdoor}}$ with m as the (identity) request to obtain the corresponding trapdoor \mathbf{t} and finally computes the new signature using the trapdoor. The reduction therefore perfectly simulates \mathcal{F} 's environment.

At the end of this game, \mathcal{F} comes up with a pair (m^*, σ^*) corresponding to a random string r^* . As we suppose that m^* is a sanitized message, there exists a message \tilde{m}^* , corresponding to a signature (σ^*, \tilde{r}^*) , which has been asked to the signing oracle, such that

$$\text{CH.Proceed}(pk_{CH}, m^*, \sigma^*, r^*) = \text{CH.Proceed}(pk_{CH}, m^*, \tilde{m}^*, \tilde{r}^*).$$

Eventually, $(m^*, \tilde{m}^*, r^*, \tilde{r}^*, \sigma^*)$ is a collision for the chameleon hash function. The final success probability and running time are straightforward, and this concludes the proof. \square

Theorem 2 (Indistinguishability). *The following distributions are perfectly indistinguishable for all \mathcal{P} , pk , sk :*

$$\begin{aligned} \mathcal{D}_{\text{Sanit}} = \{ & \tilde{\sigma} : \tilde{\sigma} = \text{Sanit}(\mathcal{P}, m, \sigma, pk, \tilde{m}, I, \mathbf{t}), (m, \tilde{m}) \in \mathcal{M}, \\ & (\sigma, I) = \text{Sign}(\mathcal{P}, m, sk), \mathbf{t} = \text{Trapdoor}(\mathcal{P}, m, sk) \} \end{aligned}$$

and

$$\mathcal{D}_{\text{Sign}} = \{ \sigma : (\sigma, I) = \text{Sign}(\mathcal{P}, m, sk), m \in \mathcal{M} \}.$$

Proof. Let's first consider $\mathcal{D}_{\text{Sign}}$. As described in the previous section, the output of Sign consists in a signature σ and a set I . The signature σ is composed of the outputs of Σ .Sign and a random (uniformly chosen) value $r_i \in \mathcal{R}$ where Sign is a classical signature scheme. Let us denote $\text{Im}(\Sigma$.Sign) the distribution of all outputs of the Sign algorithm of the chosen signature scheme. Consequently, $\mathcal{D}_{\text{Sign}} = \{ (\sigma, r_i) : \sigma \in \text{Im}(\Sigma$.Sign), $r_i \in \mathcal{R} \}$.

Let us now consider $\mathcal{D}_{\text{Sanit}}$. As described in the previous section in the description of the Sanitize algorithm, the signature $\tilde{\sigma}$ is composed of the value σ such that $(\sigma, I) = \text{Sign}(\mathcal{P}, m, sk)$ and a value \tilde{r}_i that is output by a call to the CH.Forge algorithm. As explained above, it implies that $\sigma \in \text{Im}(\Sigma.\text{Sign})$ and that σ is on the same distribution $\text{Im}(\Sigma.\text{Sign})$ as the previous one. Moreover, as shown in the previous section in useful tool, a secure identity-based chameleon hashing function has the property that $\text{Im}(\text{CH.Forge})$ is the set of all $r_i \in \mathcal{R}$ informally distributed and thus the set \mathcal{R} . Consequently,

$$\mathcal{D}_{\text{Sanit}} = \{(\sigma, r_i) : \sigma \in \text{Im}(\Sigma.\text{Sign}), \tilde{r}_i \in \mathcal{R}\}.$$

These two distributions are obviously indistinguishable in the sense of the information theory which concludes the proof. \square

3.4 An Example of Instantiation

In [2], the authors propose the following construction of an Id-based chameleon hash function.

- CH.Setup: $n = pq$ is an RSA modulus where p and q are of size the security parameter k , v is a random prime element and w corresponds to the inverse of v modulo $\varphi(n)$. Then, $sk_{CH} = (p, q, w)$ and $pk_{CH} = (n, v)$.
- CH.Extract: from the secret key sk_{CH} and an identifier Id , it is possible to construct the extracted key by first computing $J = \text{EMSA} - \text{PSS}(Id)$ and $\mathbf{t} = J^w \pmod{n}$.
- CH.Proceed: from the public key pk_{CH} , the identifier Id and a message m , choosing at random r , it is possible to compute $J = \text{EMSA} - \text{PSS}(Id)$ and the value h is then $h = J^{\mathcal{H}(m)r^v} \pmod{n}$ where \mathcal{H} is a classical hash function.
- CH.Forge: this algorithm, taking on input pk_{CH} , Id , \mathbf{t} , a new message m' , r , h and m outputs the random value r' associated with m' and h . This is done by $r' = r\mathbf{t}^{\mathcal{H}(m) - \mathcal{H}(m')} \pmod{n}$.

We can thus use this instantiation of an Id-based chameleon hash function in our proposal. The proof of Theorem 1 of [2] can be easily modified to prove that the scheme reaches the collision resistance property as defined previously in this paper. Other choices for secure identity-based chameleon hash function can be found in [17].

A lot of classical signature schemes can be used for our generic construction and one possible choice is RSA with EMSA-PSS padding [14].

4 Application to Group Content Protection

4.1 Classical Approach and Limitations

As defined in OMA DRM, a DRM agent \mathcal{A} needs an encryption key pair (sk_a, pk_a) to interact with a License Server \mathcal{L} which will provide him a license. The License Server needs a signature key pair (sk_l, pk_l) . A protected content is always encrypted with a unique secret key denoted by CEK . A license is related to a single protected content denoted by Id_c . It consists in the following fields:

- the license identifier Id_l which is a unique serial number of the license,
- the content identifier Id_c ,
- the **rights** which describe what is possible to do (read, copy, etc.) with this content using this license,
- the content key CEK encrypted for the receiver using pk_a ,
- the signature σ from the License Server of all the previous fields.

Consequently, the license can be viewed as the concatenation of a message $m = Id_l || Id_s || \text{rights} || [CEK]_{pk_a}$ and a signature σ .

There are several steps to describe the way a DRM agent is able to read a protected content in the classical approach. These are the following ones.

- **SystemCreation**: the License Server generates his signature keys.

$$(sk_l, pk_l) \leftarrow \text{SystemCreation}(\mathcal{P}).$$

In the OMA DRM standard, it is recommended to use the RSA signature scheme with an EMSA-PSS padding.

- **AgentCreation**: a DRM agent generates his encryption keys.

$$(sk_a, pk_a) \leftarrow \text{AgentCreation}(\mathcal{P}).$$

The OMA DRM standard recommends to use the RSA encryption scheme.

- **ContentEncapsulation**: during this phase, the License Server encrypts the content C using a randomly generated secret key CEK . The encrypted content is then published.

$$PC \leftarrow \text{ContentEncapsulation}(CEK, C).$$

- **LicenseGen**: this is an interactive protocol between a DRM agent \mathcal{A} and the License Server \mathcal{L} where \mathcal{A} first sends to \mathcal{L} the chosen content Id_c and its encryption public key pk_a . \mathcal{L} then creates the license $L = (m, \sigma)$ as described above and sends it back to \mathcal{A} .

$$L \leftarrow \text{LicenseGen}(Id_l, Id_c, \text{rights}, CEK, sk_l, pk_a).$$

- **ContentRead**: the DRM agent retrieves the protected content and the license, verifies the signature of the License Server on the license, verifies the **rights**, decrypts CEK using its private decryption key and finally decrypts the content using CEK .

$$C \leftarrow \text{ContentRead}(PC, L, sk_a, pk_l).$$

On one hand, the problem is that the license is completely related to the DRM agent and this latter cannot send a received license to other DRM agents. In fact, this security property is very useful in many cases since it prevents fraud, but in the context of a group of DRM agents which wants to buy some contents and then to share them, this may be a too much important restriction.

On the other hand we do not want to modify *the way a DRM agent proceeds* when reading a protected content. Consequently, it is necessary to design a way to transfer a license to another DRM agent without modifying the structure of a license nor the signature from the License Server. A license bought from the License Server and a license coming from another DRM agent should have the same structure, including the same signature. By this way, constructors need to implement only one type of DRM agent. But, for this purpose, we need first to introduce some new procedures to the ones described above.

4.2 Some New Procedures

We use the same model as the one of classical OMA DRM one we describe above except that the License Server needs a new key pair (sk, pk) related to the computation of a trapdoor that will permit a designated DRM agent to modify a designated license so that other chosen DRM agents will be able to use the license. We thus add the new following procedures.

- **TrapdoorGen**: this is an interactive protocol between a DRM agent \mathcal{A} and the License Server \mathcal{L} where the latter gives to \mathcal{A} the possibility to modify some part of a specific license.

$$\mathbf{t} \leftarrow \text{TrapdoorGen}(L, sk).$$

- **TransferLicense**: a DRM agent sends to another one a license. This latter can be used classically by the receiver using the **ContentRead** procedure.

$$\tilde{L} \leftarrow \text{TransferLicense}(L, \mathbf{t}, \widetilde{pk}_a, pk).$$

Note that we want to give the buyer a maximum of flexibility. This implies the possibility for her to choose, whenever she wants, a classical license with no possibility of transfer and later the possibility to transfer a previously obtained license. This is done by using our trapdoor sanitizable signature scheme as described in the next section. Note also that this is for this particular reason that the Ateniese et al. [1] proposal of sanitizable signature is not suitable here.

4.3 General Description

In our proposal, we maintain previous known procedures as they are. We do not modify the **SystemCreation**, **AgentCreation**, **ContentEncapsulation**, **LicenseGen** and **ContentRead** procedures. The only modification concerns the signature scheme since we do not use a classical RSA signature scheme but a trapdoor sanitizable signature scheme based on RSA with EMSA-PSS padding.

Consequently, the **SGen** call in the **SystemCreation** procedure is replaced in the OMA DRM standard by the execution of the **KeyGen** of a trapdoor sanitizable signature scheme that outputs (sk, pk) .

Moreover, in the **LicenseGen** protocol, the Σ .**Sign** algorithm is replaced by an execution of the **TSS.Sign** of the trapdoor sanitizable signature scheme. Consequently, a license has the same fields as a classical one except that the signature

from the License Server is not a classical signature. Notice however that the underlying signature scheme is standard since we can use the RSA signature scheme as a building block (as a hash and sign type signature scheme). The verification of the signature done by a DRM Agent when trying to read a content, initially done by the Σ .Verif algorithm of a classical signature scheme is also replaced by a call to the TSS.Verif algorithm of the trapdoor sanitizable signature scheme. Let us now describe the two new procedures.

- **TrapdoorGen**: this is done by executing the TSS.Trapdoor algorithm of the trapdoor sanitizable signature scheme with the public parameters, the license L and the secret key sk of the trapdoor sanitizable signature scheme as inputs. The output is the trapdoor \mathbf{t} .
- **TransferLicense**: a DRM Agent having a valid license and a trapdoor \mathbf{t} can execute the TransferLicense procedure by executing the TSS.Sanitize algorithm of the trapdoor sanitizable signature scheme with the initial license L (containing the message M and the signature σ), the trapdoor sanitizable signature public key pk , the new message \tilde{m} that corresponds to the different fields (except the signature) of the new license, the corresponding set I of indices in the message that are sanitizable (see Section 4.4 below) and the trapdoor \mathbf{t} as inputs. The output is a signature $\tilde{\sigma}$ which is concatenated to the message \tilde{m} to create the new license \tilde{L} .

4.4 On the Modification of a License

As described above, it is possible for a DRM Agent to modify some fields of a valid license using the trapdoor sanitizable signature scheme properties. We consider that a license is a (reduced) message m and the signature σ of the License Server. The message m is divided into several blocks: $m_1 = Id_l$, $m_2 = Id_c$, $m_3 = \text{rights}$ and $m_4 = [CEK]_{pk_a}$. In the following, we study which parts of the license are sanitizable.

The messages m_1 and m_2 should of course not be modified. On the contrary, the message m_4 will be modified by the DRM Agent since the receiver should be able to decrypt the CEK to read the content. Thus the value 4 necessarily belongs to the set I that is output by the TSS.Sign algorithm.

The case of the `rights` is a bit more complicated and there are several possibilities. Either it is not possible for the DRM Agent to modify the `rights`, or it is possible but only in a set of predefined values (at least no more `rights` than the ones the DRM Agent already has). In the first case, our trapdoor sanitizable signature scheme can be used as it is. In the second case, we need to modify it and it is an open problem to adapt the techniques of [9] to our scheme.

5 Conclusion

We formally introduce a new variant of sanitizable signatures and apply our new tool to manage licenses for digital contents protection within a group. We hope

that our new variant can also be useful for medical applications or secure routing (see [11]). Among some open problems, we suggest to add a traitor tracing layer or to have a better control on the message that can be sanitized.

Acknowledgments. We are grateful to anonymous referees for their valuable comments.

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and application. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
3. Ateniese, G., de Medeiros, B.: On the Key Exposure Problem in Chameleon Hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)
4. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
5. Chaum, D., van Antwerpen, H.: Undeniable Signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
6. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), 281–308 (1988)
7. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: PIATS: A Partially Sanitizable Signature Scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)
8. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
9. Klonowski, M., Lauks, A.: Extended Sanitizable Signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
10. Krawczyk, H., Rabin, T.: Chameleon Signatures. In: Proc. NDSS 2000, pp. 143–154. The Internet Society (2000)
11. Miyazaki, K., Iwamura, M., Matsumoto, T., Sakai, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. *IEICE Trans. on Fundamentals* E88-A(1), 239–246 (2005)
12. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital Documents Sanitizing Problem. IEICE technical report, ISEC 2003-20 (2003)
13. Naccache, D., Whelan, C.: 9/11: Who Alerted the CIA (And Other Secret Secrets). In: Eurocrypt 2004 rump session (2004)
14. RSA Labs. RSA Cryptography Standard: EMSAPSS – PKCS#1 v2.1 (2002)
15. Steinfeld, R., Bull, L., Zheng, Y.: Content Extraction Signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 286–304. Springer, Heidelberg (2002)
16. Suzuki, M., Isshiki, T., Tanaka, K.: Sanitizable Signature with Secret Information. Tokyo Institute of Technology Research Report, C-215, pp. 1–20 (2005)
17. Zhang, F., Safavi-Naini, R., Susilo, W.: ID-Based Chameleon Hashes from Bilinear Pairings. *Cryptology ePrint Archive, Report*, 2003/208 (2003)

Multi-factor Authenticated Key Exchange

David Pointcheval and Sébastien Zimmer

ENS – CNRS – INRIA, Paris, France
{pointcheval,zimmer}@di.ens.fr

Abstract. In order to increase the security for authenticated key exchange protocols, various authentication means can be used together. In this paper, we introduce a security model for multi-factor authenticated key exchange, which combines a password, a secure device, and biometric authentications. We thereafter present a scheme, that can be proven secure, in the random-oracle model.

1 Introduction

1.1 Motivation

Authentication is definitely one of the most important goal of modern cryptography. In order to avoid mistakes and impersonations during access control we can use various authentication means, possibly all together, that uniquely identify someone: a secret information, a biometric or user’s belongings are the most well-known examples of such authentication factors for human beings. They represent the three classes of human authentication factors generally admitted, namely:

- something you *know* (as a secret password),
- something you *have* (as an unclonable secure device with a secret key),
- something you *are* (as a biometric).

Brainard *et al.* [15] have recently proposed a fourth authentication means: *someone you know*, also called the social networking. However we focus in this paper on the classical “three-factor authentication” technique, involving the three above factors. They are all subject to various types of attacks, notably attacks that cannot be avoided using cryptographic techniques only, but require external security protections:

- the password can be recovered through social engineering (phishing [29] or malwares), and thus users have to be careful when they enter it;
- the device can be stolen, open or cloned, and thus the device must be protected using tamper-resistant techniques;
- the biometric can be copied, and thus the sensor has to be able to correctly detect whether the controlled biometric is a real one, corresponds to the human-being under control, and to certify it.

Combining the three factors in the same authentication protocol could increase the security since the adversary would have to break the three protections in order to win. However, involving the three factors does not necessarily requires the adversary to break all the protections in order to break the scheme, if the latter is not well designed: a security model for authentication based on a secret key, on a password and on a biometric, all together, has to be provided, in order to be able to formally prove that the design is correct.

In addition to simple authentication (access control), in case of success, the two parties may be interested in coming up with a common ephemeral secret key to establish a secure channel [8,17,19]. We are thus interested, not only in *authentication*, but in *Authenticated Key Exchange* [4,8]. In the following we focus on such AKE protocols, combining the three above authentication means. This basically means that if the three authentication verifications simultaneously succeed, then the 2 parties should come up with a session key that is semantically secure (indistinguishable from a truly random key to any other party), otherwise nobody learns anything.

Issues raised by PKI-based [8,17,18] and password-based [6,14,16] AKE are now well understood, and several solutions are known. The PKI/public-key setting is definitely the easiest case, since signatures [28] can be used to authenticate the flows, or alternatively the ability to decrypt, using an asymmetric encryption scheme [26,31]. In the password-based setting, one has to take care of the (off-line) dictionary attacks [9]. We indeed cannot avoid the *on-line* dictionary attack, which consists in trying to impersonate one party with a random password, and do it again, until the correct password is used. We thus want to prove that this is the best attack. Note that in many cases, such attacks can be prevented or damages can be reduced with appropriate techniques (delays after a failure, limited number of failures, etc).

However, biometric-based authentication raises quite different issues. First of all, biometric cannot be assumed a secret information. Indeed, recovering a fingerprint from the object someone has just touched is an easy task, or getting an image of the iris simply requires a camera. That is why considering biometric as a truly secret information and treating it the same way as a private key is not reasonable in practice, even if this scenario has often been assumed in the literature [30,24,12,13,23].

On the other hand, if the biometrics are public, how do we prevent an adversary from impersonating an honest user? The only way to use biometrics for authentication is to guarantee that the biometric template comes from a real living human being and not from a fake copy. Several technical solutions have been elaborated to guarantee this (authenticated channels, various biometric features controlled at the same time, sensor under human supervision, ...). The assumption that biometrics really come from the living human being under control is called the *liveness assumption*. It also implies that all computations made from the biometric data are done honestly. This assumption is not only useful for authentication, it is compulsory to ensure authentication security. This is a

strong, but necessary, assumption. Practical solutions exist to achieve, but are out of the scope of the this paper.

Secondly, and more importantly from a technical point of view, two measurements of the same biometric lead to different templates. Since a specific template cannot be reproduced, a matching mechanism has to be used to compare two templates and determine if they come from the same biometric or from two different persons. The matching can for example be based on a simple threshold on the distance between the candidate template and the reference template (as it is the case for iris [20]). However all known matching systems are not foolproof: they introduce two possible errors, “false acceptance” (when the system accepts someone it should not) and “false rejection” (when the system does not recognize someone it should). Therefore, an AKE protocol based on biometrics has to deal with these measurement errors and make this matching possible, but should not increase significantly the “false acceptance” and “false rejection” rates.

Finally, biometrics can be used to unequivocally identify an individual and are often linked with other personal information in the database. Since these databases can be vulnerable to internal or external adversaries, the privacy of the database is a classical requirement. Even if we already noticed that they cannot be considered as private information, biometric templates are critical data, especially when they are gathered in a database. Privacy is thus a major concern here.

1.2 Related Works

As already mentioned, literature about PKI-based and password-based AKE is rich of many results [8,17,18,6,14,16].

Dealing with biometric measurement errors is a much more challenging task and two dedicated tools were formalized by Dodis *et al.* [24]: secure sketches and fuzzy extractors. They allow, from an erroneous biometric measurement and public information, to always generate the same biometric template and random bitstring respectively. These tools were improved and allowed to design biometric-based AKE [24,12,13]. However, these tools rely on the assumption that biometrics are private information, which we do not allow in this paper.

Several efforts were taken to design authentication protocols where the matching is made on the client side [3]. But in client-side protocols the client sensor must record a reference biometric template for the user(s), which can be heavy if numerous people use the same sensor.

Despite all the efforts taken for 1-factor authentication or AKE protocols, literature does not tell much on multi-factor authentication protocols. In [11], an encoding for fingerprints is proposed, which is thereafter included in the design of a two-factor authentication protocol. Their fingerprint encoding has the property that two measurements of the same fingerprint leads to the same encoding, despite the errors. They make good use of it, since no matching is needed anymore and they can use classical cryptographic tools. They propose to use zero-knowledge proofs of knowledge [27], so that the database cannot have any information about the biometric template that it records. They assume that

the biometric is private, however their protocol can be proven secure even if biometric template is public. This protocol has nice features, but it heavily relies on the fact that thanks to their encoding, they get rid of errors. There is not such encoding for all biometrics and this protocol is therefore very restrictive. Furthermore, it achieves authentication only, but does not help to establish a secure channel.

1.3 Our Solution

We propose a Multi-Factor based AKE (MFAKE) which preserves database privacy. From three factors (a password, a high entropy secret key and a biometric template) the protocol generates a common semantically secure secret key, in order to establish a secure channel. The protocol is designed so that the matching is made on the server side and is adapted for a matching based on a simple threshold on the distance between the candidate template, and a reference template. Therefore, it is particularly well-suited to iris which is efficiently encoded on 1024-bit string, but can also apply on some other biometric techniques, with appropriate encoding.

Derived from PKI-AKE, PAKE and biometric-based AKE security models, we first define a new and clear security model for MFAKE protocols, which combines all the corresponding security properties. We chose to extend the Real-or-Random model, since the latter is strictly stronger than the Find-then-Guess model in the password-based setting [1]. The model allows the adversary to make several corruptions, on the secret key, the password, or the sensor. And despite two corrupt queries, the new keys should still remain semantically secure: in this model a protocol is provably as secure as the strongest remaining factor. Furthermore, our model also deals with the forward-secrecy, which means that, even when all the authentication means are corrupted, a session key established before the last corruption remains semantically secure. However, note that we only consider client-authentication (Test-queries will be allowed to the server only, in the formal security model below). This can be seen as a strong limitation, but it is not in practice: if the password and the secret keys are compromised, an adversary can easily play the role of the server, since there is no more secret (the biometric is public and the liveness assumption is valid on the client side only), whatever the protocol is. Authentication of the server to the client could be satisfied, until the two secret information related to the client (secret key and password) are compromised, but we do not address it in this model.

Then we also provide a protocol that is secure, according to this model, in the random-oracle [7]. This protocol records an encrypted version of the biometric template on the server side. Therefore privacy of the database (and thus of all the biometric templates) is preserved, even to the server, and thus even if the server is compromised. The protocol is proven to have a tight security proof: when the password is the last factor not to be corrupted, on-line dictionary attacks are the most efficient attacks that can be mounted; when the biometric is the last one, the adversary probability to be accepted is nearly equal to the false-acceptance

probability; when the secret key is still private, the security level is quite strong since it requires the adversary to break the Diffie-Hellman problem [22].

2 Security Model

In this section, we describe the security model for multi-factor authenticated key exchange (later denoted MFAKE). This model is built upon the usual password-authenticated key exchange security model [8,6], in the Real-or-Random indistinguishability framework [5,11].

2.1 Notation

We first explain the notation and the assumptions about the authentication means.

PARTICIPANTS, SESSIONS AND PARTNERING. In a MFAKE, participants are either clients \mathcal{C} or a unique, trusted server S . The server and every client can activate several instances at a time, in order to run several sessions concurrently. The instance i of the entity U , where U is a client or the server, is denoted as Π_U^i . This instance includes three variables, initialized as *null*:

- pid_U^i : the *partner identifier* which is the instance with whom Π_U^i believes it is interacting,
- sid_U^i : the *session identifier*, in practice it can be the transcript seen by Π_U^i (concatenation of the received/sent flows, excepted the last one).
- acc_U^i : a boolean variable which is fixed at the end of the session and denotes whether the instance Π_U^i goes in an *accepted state* or not.

The two instances Π_U^i and $\Pi_{U'}^j$ are said to be *partners* if the following conditions are fulfilled:

1. $\text{pid}_U^i = \Pi_{U'}^j$, and $\text{pid}_{U'}^j = \Pi_U^i$;
2. $\text{sid}_U^i = \text{sid}_{U'}^j \neq \text{null}$;
3. $\text{acc}_U^i = \text{acc}_{U'}^j = 1$.

LONG-LIVED KEYS. Each client \mathcal{C} owns a tuple $t_{\mathcal{C}} = (\mathcal{D}_{\mathcal{C}}, \text{sk}_{\mathcal{C}}, \text{pwd}_{\mathcal{C}})$, where $\mathcal{D}_{\mathcal{C}}$ is a probability distribution for his biometric, while $\text{sk}_{\mathcal{C}}$ and $\text{pwd}_{\mathcal{C}}$ are a high-entropy private key and a low-entropy password respectively. The server holds a list of tuples $t_S = \langle t_S[\mathcal{C}] \rangle$, where $t_S[\mathcal{C}]$ is a transformed-tuple of $t_{\mathcal{C}}$. More precisely, when the client \mathcal{C} enrolls in the system, he generates a biometric template $W_{\mathcal{C}}$, according to the distribution $\mathcal{D}_{\mathcal{C}}$, as well as two private data $\text{sk}_{\mathcal{C}}$ and $\text{pwd}_{\mathcal{C}}$. The tuple $t_S[\mathcal{C}]$ is then an (injective) transformation of $(W_{\mathcal{C}}, \text{sk}_{\mathcal{C}}, \text{pwd}_{\mathcal{C}})$.

BIOMETRIC TEMPLATES. As explained above, for each client \mathcal{C} , $\mathcal{D}_{\mathcal{C}}$ defines the probability distribution of his biometric (fingerprint, face, iris, etc). In order to be relevant for authentication, we have to make some assumptions about the matching process, and more precisely about the encoding and the Hamming distance, since we will use this distance in the matching decision:

- on the one hand, the distance between two templates W_C and W'_C of the same biometric is low with great probability. More concretely, there is a threshold t , such that for any C ,

$$\Pr[W_C \leftarrow \mathcal{D}_C, W'_C \leftarrow \mathcal{D}_C : d_H(W_C, W'_C) \leq t] \geq 1 - \varepsilon_{fr}.$$

The subscript fr stands for “false rejection”.

- on the other hand, for any pair of distinct clients $C \neq C'$, the distance between W_C and $W_{C'}$ is high with great probability. More precisely, there exist a threshold $\tau \geq t$, such that for any $C \neq C'$,

$$\Pr[W_C \leftarrow \mathcal{D}_C, W_{C'} \leftarrow \mathcal{D}_{C'} : d_H(W_C, W_{C'}) > \tau] \geq 1 - \varepsilon_{fa}.$$

The subscript fa stands for “false acceptance”.

We assume that for all the clients C , the biometric distribution \mathcal{D}_C is public. Under the liveness assumption explained below the biometric acceptance will guarantee that the client *is* like the intended client.

PRIVATE DATA. The private key component sk_C is chosen uniformly in a set of private keys *Keys*, where *Keys* is assumed to be very large (with high entropy), such that $1/\#Keys$ is negligible. It will be stored in a secure device. The acceptance of this private key, with respect to a public key, will guarantee that the client *has* the device.

On the opposite, the password component pwd_C is chosen in a fixed low-entropy dictionary $Dict \subset \mathbb{Z}_p^*$, according to the probability distribution \mathcal{D}_{pwd} . We denote by $\mathcal{D}_{pwd}(q)$ the sum of the probability of the q most probable passwords according to \mathcal{D}_{pwd} . The knowledge of the password will guarantee that the client *knows* it.

LIVENESS ASSUMPTION. Since we assume the biometric to possibly be public (the opposite assumption is not reasonable in practice), then the *liveness assumption* [32,21], though quite strong, is necessary. It prevents the attacker from making replay attacks and from altering the computations made by the sensor. The liveness assumption implies that the biometric is fresh, comes from a real living person (and not using a fake biometric feature), and that the computations are made from this biometric honestly.

To model this assumption, a computation oracle $\text{Compute}(\Pi_C^i, W', sk, pwd)$ is used: according to the state of the client instance Π_C^i , from the secrets sk, pwd and a random value of W' , it computes honestly the message which would have been generated by C with these inputs, following the protocol.

As it models an attempt of the attacker to authenticate using its own biometric, W' has to be chosen according to a (wrong) distribution \mathcal{D} , such that $\Pr[W' \leftarrow \mathcal{D}, W_C \leftarrow \mathcal{D}_C : d_H(W', W_C) > \tau] \geq 1 - \varepsilon_{fa}$.

With the *liveness assumption* for the client C , we consider that all the messages involving the biometric, claimed to be sent by C , have been previously generated by the computation oracle.

CORRUPTION. As explained below, the adversary will be allowed to corrupt a client C , by learning the password pwd_C (phishing), by getting the private key sk_C (side-channel attack on the device), or by breaking the above liveness assumption (attack on the sensor).

2.2 Semantic Security

ADVERSARIAL CAPABILITIES AND GOALS. The semantic security of the key is modeled using the Real-or-Random paradigm [5,11]. At the beginning of the game, the challenger chooses a random bit b which determines its behavior when answering **Test**-queries during the game (it provides either real keys or random keys to the adversary). The adversary may interact with protocol instances through several oracles, and at the end of the game, she sends a bit b' . If $b = b'$, she wins, otherwise, she loses. The available queries are as follows:

- **Send**(m, Π_U^i): this query allows the adversary to play with the instances, by intercepting, forwarding, modifying or creating messages. The output of this query is the answer generated by instance Π_U^i to the message m . As stated above, if $\text{pid}_S^i = \Pi_C^j$ is the client instance with whom the server believes to talk, if the liveness assumption still holds for the client \mathcal{C} (no corruption) and if the computation of m involves the biometric, then m has to have been previously generated through a **Compute**($\Pi_C^j, W', \text{sk}, \text{pwd}$) query.
- **Reveal**(Π_U^i): this query models the leakage of information about the session key agreed on by the parties. For example, if it is misused afterward. Therefore, if no session key is defined for this instance, or if the instance (or its partner) has been tested (see below), then the output is \perp . Otherwise, the oracle outputs the session key computed by the instance Π_U^i .
- **CorruptKey**(\mathcal{C}, a): this query models corruption capabilities of the adversary. She can indeed steal/break one or several authentication factors of clients.
 - If $a = 1$, the oracle outputs the password pwd_C of \mathcal{C} ;
 - if $a = 2$, the oracle outputs the secret key sk_C of \mathcal{C} ;
 - if $a = 3$, the attacker is now allowed to submit *any* message involving the biometry, without asking the computation oracle **Compute** before. It models the attack against the liveness assumption.

Note that in the following, we will restrict to non-adaptive corruptions: no corruption can be performed during a session, but before a new session starts.

To formally model the semantic security with respect to client authentication, the adversary can ask **Test**-queries, but to the server only: we are interested in the privacy of the key established with the real server only. We only consider adversaries whose goal is to impersonate a client to the server. Of course, to achieve this goal, the adversary may try to impersonate the server to the client in order to learn some information about the long-lived keys of the client. But only a client impersonation will be considered as a successful attack:

- **Test**(Π_S^i): if Π_S^i is not *fresh* (see below), then output \perp , otherwise the oracle sends
 - the session key of instance Π_S^i (that is **Reveal**(Π_S^i)), if $b = 1$ – the real case;
 - a random key from the same domain, if $b = 0$ – the random case.

FRESHNESS. The freshness notion basically defines session keys that are not trivially known to the adversary. Since we will focus on the freshness of the server only, we say that the session key of instance Π_S^i is fresh if:

- upon acceptance, \mathcal{C} (corresponding to the partner of Π_S^i) was not *fully corrupted*. This means that strictly less than 3 **CorruptKey**-queries had been asked to the client \mathcal{C} ;
- no **Reveal**-query has been sent to either Π_S^i or its partner.

SEMANTIC SECURITY. Let denote by **Succ** the event that the adversary \mathcal{A} correctly guesses the bit b used by the challenger during the above attack game. The **mfake-advantage** $\text{adv}_P^{\text{mfake}}(\mathcal{A})$ and the advantage function of the protocol P are respectively:

$$\text{adv}_P^{\text{mfake}}(\mathcal{A}) = 2 \cdot \Pr[\text{Succ}] - 1, \quad \text{adv}_P^{\text{mfake}}(t, Q) = \max_{\mathcal{A}} \{ \text{adv}_P^{\text{mfake}}(\mathcal{A}) \},$$

where the maximum is over all the attackers with time-complexity at most t and number of queries at most Q .

FORWARD-SECURITY. Forward-security means that as soon as a session key is securely generated (semantically secure), it will remain secure even after corruption. In order to capture this security level, the model must allow the adversary to perform **Test**-queries, even when the 3 **CorruptKey**-queries have been asked, but on sessions completed before the full corruption of the client. One can also consider that upon acceptance, a session is fresh if less than 3 **CorruptKey**-queries have been asked.

CLIENT AUTHENTICATION. We also usually model an attack against the unilateral authentication of the client to the server by considering sessions where the server accepts, but without any client-partner. Let denote by **Succ** the event that a server instance accepts with no partner instance of the client (with the same partial transcript).

The **auth-success** $\text{Succ}_P^{\text{auth}}(\mathcal{A})$ and the success function of the protocol P are respectively:

$$\text{Succ}_P^{\text{auth}}(\mathcal{A}) = \Pr[\text{Succ}], \quad \text{Succ}_P^{\text{auth}}(t, Q) = \max_{\mathcal{A}} \{ \text{Succ}_P^{\text{auth}}(\mathcal{A}) \},$$

where the maximum is over all the attackers with time-complexity at most t and number of queries at most Q .

3 Description of the Protocol

The complete description of our protocol is provided Figure [1](#). It assumes a common setup, with parameters (u, v, p, g, q) , where g is an element of order q in \mathbb{Z}_p^* , and generates the subgroup \mathbb{G} . Then, u and v are random elements in \mathbb{G} . We also model H as a random oracle [7](#).

The server stores all the data corresponding to user \mathcal{C} , provided during the enrollment phase:

- the public key $h = g^{x_C}$, related to the high-entropy secret x_C ;
- an El Gamal encryption [25] of the reference biometric template $W_C = (W_i)_{i \leq N}$ —where W_i is the i -th bit of W_C and N the number of bits— under the public key $h = g^{x_C}$, that is tuple of pairs $(g^{r_i}, h^{r_i} g^{W_i})_i$;
- the password $\text{pwd}_C \in \mathcal{D} \subset \mathbb{Z}_q^*$.

One can note that the server actually does not know the biometrics of the clients since they are encrypted under keys chosen by the clients.

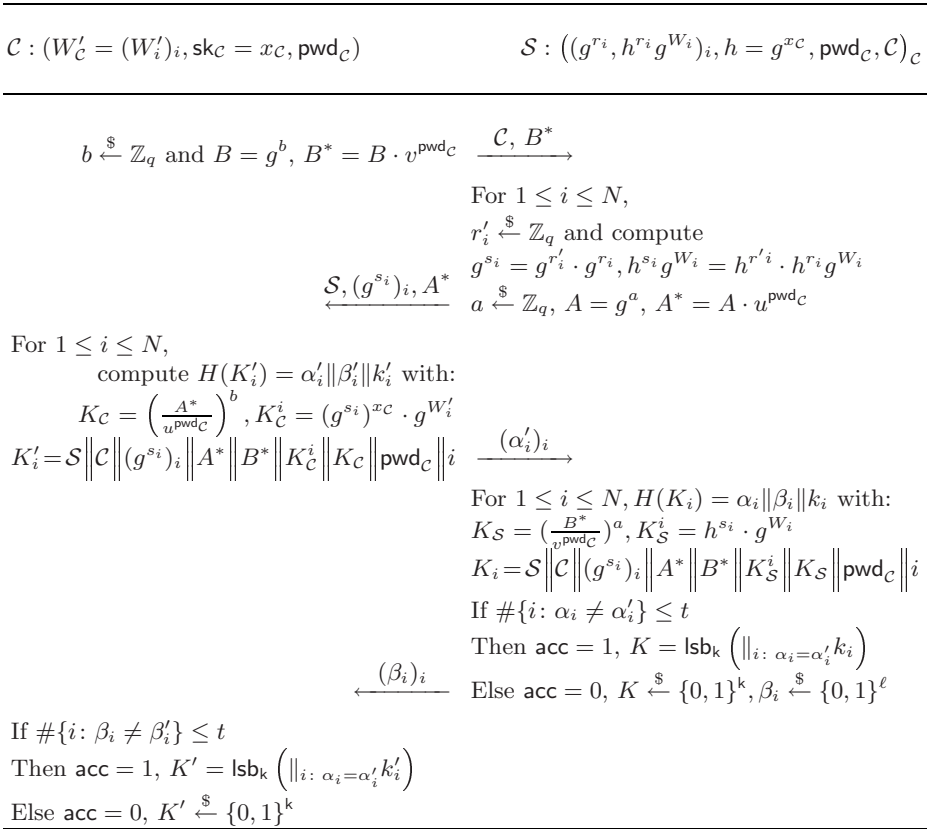


Fig. 1. Our MFAKE Protocol

For authenticating himself, the client \mathcal{C} owns an ephemeral biometric template $W'_C = (W'_i)_i$; the long-term private key x_C ; and the password $\text{pwd}_C \in \mathcal{D} \subset \mathbb{Z}_q^*$.

The protocol guarantees that, if the ephemeral template W'_C is close enough to the reference template W_C (who you are), if the private key x_C corresponds to the public key h (what you have), and if the passwords are the same (what

you know), then the server accepts the client, and they agree on an ephemeral common secret $K' = K$.

We namely want to prove that unless the three authentication factors have been corrupted, no adversary can impersonate a client to the server. And all the keys actually agreed on between a client and the server are semantically secure, even after corruptions (forward-secrecy).

Basically, (B^*, A^*) corresponds to the Password-Authenticated part (similar to EKE [9,2]); for each i , (h, g^{s_i}) is a Diffie-Hellman key agreement which leads to the key $g^{x_C \cdot s_i}$, with x_C used for authentication. Note that the s_i are rerandomized every time, so that the Diffie-Hellman key exchange is not static. The bit W_i (or W'_i for the client) of the biometric template is used as a mask and we obtain $g^{x_C \cdot s_i} \cdot g^{W_i}$ ($g^{x_C \cdot s_i} \cdot g^{W'_i}$ for the client). If for most of i the masks W_i and W'_i are equal, then for most of i the authenticators α'_i and verifiers α_i will be equal also, as well as β_i and β'_i , and k_i and k'_i .

To define the partnership in our protocol, we have to precise that the sid is equal to the first two flows $((C, B^*), (\mathcal{S}, (g^{s_i})_i, A^*))$.

4 Security of the Protocol

Before stating the security result, let us remind the computational assumption on which the security will rely.

COMPUTATIONAL DIFFIE-HELLMAN PROBLEM. Let \mathbb{G} be a cyclic group of order q . Let g be a generator of \mathbb{G} , let (x, y) be two integers uniformly chosen in \mathbb{Z}_q . The computational Diffie-Hellman problem states that, given (g^x, g^y) , it is difficult to compute $g^{xy} = \text{CDH}_g(g^x, g^y)$.

Let A be a CDH-adversary with running time at most T . We denote by $\text{Succ}_g^{\text{cdh}}(A)$ the probability that A succeeds in computing g^{xy} from (g^x, g^y) and by $\text{Succ}_g^{\text{cdh}}(T) = \max_A \{ \text{Succ}_g^{\text{cdh}}(A) \}$ where the maximum is taken over all the adversaries with running-time at most T .

BIOMETRIC. We remind that for any client C and any adversary which uses a true biometric W' , we have $\Pr[d_H(W', W_C) > \tau] \leq 1 - \varepsilon_{\text{fa}}$ where τ is an integer greater than t . The protocol does not increase the false-rejection probability but it does increase the false-acceptance probability, due to the additional check on the $\alpha_i = \alpha'_i$ equalities. The increasing is upper-bounded by

$$\Pr[\#\{i : \alpha'_i \neq \alpha_i\} \leq t \mid d_H(W', W_C) > \tau] \leq \frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}}.$$

A protocol session between two honest entities is correct if for all i , $K_i = K'_i$ is equivalent to $\alpha_i = \alpha'_i$ or $\beta_i = \beta'_i$. It fails if there is an index i such that $K_i \neq K'_i$ and $\alpha_i = \alpha'_i$ or $\beta_i = \beta'_i$. As there are at most t indexes i such that $K_i \neq K'_i$ the probability that an honest protocol session is not correct is upper bounded by $2t \Pr[\alpha_i = \alpha'_i : K_i \neq K'_i]$ which is equal to $2t/2^\ell$.

Theorem 1. *Let us consider the above protocol P over a group of prime order q , where the dictionary of passwords is equipped with the distribution $\mathcal{D} \subset \mathbb{Z}_q^*$. Let \mathcal{A} be an adversary against the semantic security within a time bound T , with less than q_{session} Send-queries and asking less than q_h queries to the random oracle. Then we have*

$$\begin{aligned} \text{adv}_P^{\text{mfake}}(\mathcal{A}) &\leq 2 \sum_c \mathcal{D}(q_c) + 4q_h^2 \cdot \text{Succ}_g^{\text{cdh}}(T + 4\tau_e) + \frac{q_{\text{session}}^2}{q} + \frac{2q_h}{q} \\ \text{Succ}_P^{\text{auth}}(\mathcal{A}) &\leq \sum_c \mathcal{D}(q_c) + 2q_h^2 \cdot \text{Succ}_g^{\text{cdh}}(T + 4\tau_e) + \frac{q_{\text{session}}^2}{2q} + \frac{q_h}{q} \\ &\quad + q_{\text{session}} \left(\varepsilon_{\text{fa}} + \frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N} (t-1)!} \right) \end{aligned}$$

where τ_e denotes the computational time for one exponentiation and q_c the number of active sessions the adversary ran against client C .

Proof. The proof consists of a sequence of games:

Game 0. This is the real attack game, against the protocol. We are interested in the two following events:

- \mathbf{S}_0 (for semantic security) which occurs if the adversary correctly guesses the bit b chosen at the beginning of the game.
- \mathbf{A}_0 (for client authentication), which occurs if a server instance accepts with no partner instance of the client (with the same transcript).

Actually in any game \mathbf{G}_n , we study the event \mathbf{A}_n , and the event \mathbf{S}_n . Note that

$$\text{adv}_P^{\text{mfake}}(\mathcal{A}) = 2 \Pr[\mathbf{S}_0] - 1, \quad \text{Succ}_P^{\text{auth}}(\mathcal{A}) = \Pr[\mathbf{A}_0].$$

Therefore

$$\begin{aligned} \text{adv}_P^{\text{mfake}}(\mathcal{A}) &= 2 \Pr[\mathbf{S}_n] - 1 + 2(\Pr[\mathbf{S}_0] - \Pr[\mathbf{S}_n]) \leq 2 \Pr[\mathbf{S}_n] - 1 + 2 \sum_{i=0}^{n-1} \Delta_i \\ \text{Succ}_P^{\text{auth}}(\mathcal{A}) &= \Pr[\mathbf{A}_n] + (\Pr[\mathbf{A}_0] - \Pr[\mathbf{A}_n]) \leq \Pr[\mathbf{A}_n] + \sum_{i=0}^{n-1} \Delta_i, \end{aligned}$$

if we denote by Δ_i the distance between games \mathbf{G}_i and \mathbf{G}_{i+1} .

Game 1. In this game, we simulate the random oracles (H , but also an additional function H' that will appear in the game \mathbf{G}_3) as usual by maintaining lists Λ_H and $\Lambda_{H'}$. We also simulate all the instances, as the real players would do, for the Send-queries and the Reveal and Test-queries. From this simulation, we see that the game is indistinguishable from the real attack: $\Delta_0 = 0$.

Note that since the probability distributions of the biometrics are public, we draw a random reference template for each client, to be used/known by the server. And when needed (simulation of a client), we can draw a random biometric according to the (public) probability distribution of the client’s biometric.

Game 2. In order to guarantee independence of the sessions, we cancel games in which some collision on the session transcripts $((\mathcal{C}, B), (\mathcal{S}, A^*, (g^{s_i})_i))$ appear. Since transcripts involve at least one honest party, $(A^*, (g^{s_i})_i)$ or B^* is truly uniformly distributed. Therefore the collision probability is upper bounded by $q_{\text{session}}^2/2q$, where q_{session} is the number of sessions: $\Delta_1 \leq q_{\text{session}}^2/2q$.

Game 3. We now replace the generation of the authenticators and session keys with a private oracle H' instead of H , for all the sessions that are fresh (which can be tested: involving a server so that the intended client is not fully corrupted), and also for all the sessions involving a client (but no server) for which the password and the secret key are unknown (none of the 1-CorruptKey and 2-CorruptKey-queries has been asked): instead of using the public oracle H , we use the private oracles H' , on K_i and K'_i computed as

$$K_i = \mathcal{S} \left\| \mathcal{C} \left\| (g^{s_i})_i \left\| A^* \left\| B^* \left\| g^{W_i} \right\| i \right. \right. \right. \quad K'_i = \mathcal{S} \left\| \mathcal{C} \left\| (g^{s_i})_i \left\| A^* \left\| B^* \left\| g^{W'_i} \right\| i \right. \right. \right.$$

As already explained, we have chosen a random reference biometric template for each user. And when needed, we can draw a random biometric according to the (public) probability distribution of the client's biometric. Thus we can include g^{W_i} or $g^{W'_i}$ in the above public computations, in order to make K_i and K'_i possibly different, even for compatible biometric templates.

We do not use none of K_C, K_C^i, K_S and K_S^i anymore, therefore we can omit their computations. Besides, we do not use A and B anymore, therefore we can change the computations of A^* and B^* by $a^* \xleftarrow{\$} \mathbb{Z}_q, A^* = g^{a^*}$ and $b^* \xleftarrow{\$} \mathbb{Z}_q, B^* = g^{b^*}$. Lastly, since we do not use neither the password nor the secret key, we can choose them at the last moment: for the password-corrupt query (1-CorruptKey) or for the secret key-corrupt query (2-CorruptKey), or at the very end of the game only (when the adversary gives her answer).

However, when a client is fully corrupted (adversary against the server) or the adversary plays against a client from which she knows the password and the secret key, the keys K_i and K'_i are computed normally and we use the public oracle H again.

Note that we restrict to non-adaptive corruptions, and thus, when a session starts, we know the corruption status of a client. Then, requests to the Compute-oracle will also focus on such a session for which we know the corruption status, since the biometric is only involved in the third round. The latter oracle indeed has to know how to perform the simulation of K'_i , using either H or H' .

The games \mathbf{G}_2 and \mathbf{G}_3 are indistinguishable unless some specific hash query is asked (for a session made before the last CorruptKey-query): if the adversary asks either

$$\mathcal{S} \left\| \mathcal{C} \left\| (g^{s_i})_i \left\| A^* \left\| B^* \left\| K_C^i \left\| K_C \left\| \text{pwd}_C \right\| i \right. \right. \right. \text{ or } \mathcal{S} \left\| \mathcal{C} \left\| (g^{s_i})_i \left\| A^* \left\| B^* \left\| K_S^i \left\| K_S \left\| \text{pwd}_C \right\| i \right. \right. \right.$$

for some transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$, and some index i , to the H function, whereas the H' function has been used by the simulator. We denote by AskH_3 such an event. Note that, it can be decided whether this event happened only when the password and the secret key have both been chosen.

In this game, for all clients, the $(\alpha_i)_i$, the $(\beta_i)_i$ and the key are computed from a private random oracle. Therefore, whatever the bit b involved in the Test-query, the answer is random, and independent for all the sessions, unless some transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ appeared twice, but this has already been excluded in game \mathbf{G}_2 . Therefore we have:

$$\Delta_2 \leq \Pr[\text{AskH}_3] \quad \text{and} \quad \Pr[\mathbf{S}_3] = \frac{1}{2}.$$

Similarly, the only possibility for the adversary to authenticate against a true server instance is to guess the α_i at random or to use the Compute-oracle, unless some transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ appeared twice.

If she tries to guess the α_i at random, since $|\alpha_i| = \ell$, then her probability to succeed is upper-bounded by:

$$\frac{1}{2^{N\ell}} \cdot \sum_{k=0}^t \binom{N}{k} (2^\ell - 1)^k \leq \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N} (t - 1)!},$$

If she uses the Compute-oracle, all the α'_i and β_i are generated through a trusted computation oracle and since the adversary uses her own biometric W' , which is, with high probability, quite different from the client biometric $W_{\mathcal{C}}$, her probability to succeed is exactly the false-acceptance probability computed earlier.

As a consequence,

$$\Pr[\mathbf{A}_3] \leq q_{\text{session}} \left(\varepsilon_{\text{fa}} + \frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N} (t - 1)!} \right).$$

Game 4. Our goal is now to upper-bound the probability of event AskH₃. We denote by AskH₄ the same event in this game and have AskH₃ ≤ AskH₄ + Δ₃. In this game, we receive a Diffie-Hellman pair $(X = g^x, Y = g^y)$, and we will try to show that the probability of event AskH is related to the probability of computing the Diffie-Hellman value of (X, Y) . We set $u = X$ and $v = Y$. We furthermore cancel games in which, for a transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$, which both

- was generated before a password-corrupt query to the client \mathcal{C} was made
- comes from an execution involving the adversary, against either an instance of the client \mathcal{C} or the server \mathcal{S}

there are two tuples $(A^*, B^*, \text{CDH}_g(A^*/u^{\text{pwd}_k}, B^*/v^{\text{pwd}_k}), i_k)$, with two different passwords pwd_0 and pwd_1 and two, possibly different, indexes i_0 and i_1 , such that

$$\mathcal{S} \parallel \mathcal{C} \parallel (g^{s_i})_i \parallel A^* \parallel B^* \parallel K_S^{i_k} \parallel \text{CDH}_g(A^*/u^{\text{pwd}_k}, B^*/v^{\text{pwd}_k}) \parallel \text{pwd}_k \parallel i_k$$

is in Λ_H .

DISTANCE. We first easily show that $\Delta_3 \leq q_h^2 \cdot \text{Succ}_g^{\text{cdh}}(T + 3\tau_\varepsilon)$. To this aim, we remind that the distance we study comes from the fact we have canceled games

in which, for some specific transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ —which was generated before a password-corrupt query to the client \mathcal{C} was made and which comes from an execution involving the adversary, against either an instance of the client \mathcal{C} or the server \mathcal{S} —, there are two tuples $(A^*, B^*, \text{CDH}_g(A^*/u^{\text{pwd}_k}, B^*/v^{\text{pwd}_k}), i_k)$, with two different passwords pwd_0 and pwd_1 and two, possibly different, indexes i_0 and i_1 , such that

$$\mathcal{S} \parallel \mathcal{C} \parallel (g^{s_i})_i \parallel A^* \parallel B^* \parallel K_S^{i_k} \parallel \text{CDH}_g(A^*/u^{\text{pwd}_k}, B^*/v^{\text{pwd}_k}) \parallel \text{pwd}_k \parallel i_k$$

is in Λ_H .

If such a pair exists, then for $k = 0, 1$: $\text{CDH}_g(A^*/u^{\text{pwd}_k}, B^*/v^{\text{pwd}_k})$ is equal to

$$\frac{\text{CDH}_g(A^*, B^*) \cdot \text{CDH}_g(u^{-1}, B^*)^{\text{pwd}_k} \cdot \text{CDH}_g(A^*, v^{-1})^{\text{pwd}_k}}{\text{CDH}_g(u, v)^{\text{pwd}_k^2}}.$$

Since we simulated either A^* or B^* , knowing the discrete logarithms, we can extract $\text{CDH}_g(X, Y)$. Let us show it when $B^* = g^{b^*}$, it works similarly when we know $A^* = g^{a^*}$: since the two passwords are different and non-zero,

$$\text{CDH}_g(X, Y) = \frac{\text{CDH}_g(A^*/u^{\text{pwd}_0}, B^*/v^{\text{pwd}_0})^{1/\text{pwd}_0(\text{pwd}_1 - \text{pwd}_0)}}{\text{CDH}_g(A^*/u^{\text{pwd}_1}, B^*/v^{\text{pwd}_1})^{1/\text{pwd}_1(\text{pwd}_1 - \text{pwd}_0)}} \cdot (A^*)^{-b^*/\text{pwd}_0\text{pwd}_1}.$$

CONCLUSION. In order to conclude with the computation of $\text{Pr}[\text{AskH}_4]$, we distinguish the events when the transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ comes from an execution between:

- two instances of \mathcal{C} and \mathcal{S} , or an instance of \mathcal{C} or \mathcal{S} and the adversary but the flows are all oracle-generated, this event is denoted by AskH-Passive_4 ;
- an instance of \mathcal{C} and the adversary, where at least one flow is not oracle-generated, this event is denoted by AskH-withC_4 ;
- an instance of \mathcal{S} and the adversary, where at least one flow is not Compute-oracle-generated, this event is denoted by AskH-withS_4 ;

Assume that there is a tuple $(A^*, B^*, D = \text{CDH}_g(A^*/u^{\text{pwd}}, B^*/v^{\text{pwd}}))$ such that

$$\mathcal{S} \parallel \mathcal{C} \parallel (g^{s_i})_i \parallel A^* \parallel B^* \parallel K'_C \parallel D \parallel \text{pwd} \parallel i \text{ or } \mathcal{S} \parallel \mathcal{C} \parallel (g^{s_i})_i \parallel A^* \parallel B^* \parallel K'_S \parallel D \parallel \text{pwd} \parallel i$$

is in Λ_H , for any password pwd of the adversary’s choice.

If the corresponding transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ comes from an execution between instances of \mathcal{C} and \mathcal{S} , it means that both A^* and B^* have been simulated (and the adversary was only passive). In this case, we know the discrete logarithms a^* and b^* , and

$$\text{CDH}_g(A^*/u^{\text{pwd}}, B^*/v^{\text{pwd}}) = \frac{g^{a^*b^*} \cdot (v^{a^*} u^{b^*})^{\text{pwd}}}{\text{CDH}_g(v, u)^{\text{pwd}^2}}.$$

Since pwd is non-zero in \mathbb{Z}_q , it can be inverted modulo q and then,

$$\text{CDH}_g(X, Y) = \left(\frac{g^{a^*b^*} \cdot v^{a^* \cdot \text{pwd}} \cdot u^{b^* \cdot \text{pwd}}}{\text{CDH}_g(A^*/u^{\text{pwd}}, B^*/v^{\text{pwd}})} \right)^{1/\text{pwd}^2}.$$

Therefore $\text{Pr}[\text{AskH-Passive}_4] \leq q_h \times \text{Succ}_g^{\text{cdh}}(T + 4\tau_e)$.

If the corresponding transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ comes from an execution between an instance of \mathcal{C} and the adversary, where at least at one flow is not oracle-generated, it means that B^* has been simulated and the other has been generated by the adversary. We know that either the secret key-corrupt query or the password corrupt query has not been asked (otherwise the simulation was performed using H in game \mathbf{G}_3).

- Assume that the secret key-corrupt query has not been made before this session, then x_c and h are unknown to the adversary. Then it is quite hard to compute $h^{s_i} = (g^{s_i})^{x_c}$ (no information at all): q_h/q .
- If the secret key-corrupt query has been made, it implies that the password-corrupt query has not been made. Due to the games which were canceled in this game, there is at most one password pwd such that there exists an index i , $1 \leq i \leq N$, such that:

$$\mathcal{S} \parallel \mathcal{C} \parallel (g^{s_i})_i \parallel A^* \parallel B^* \parallel K_S \parallel K_S \parallel \text{pwd} \parallel i$$

is in Λ_H . In other words, for every transcript, there is only one password which can be tested by the adversary: $\sum_{\mathcal{C}} \mathcal{D}(q_{\mathcal{C}})$.

If the corresponding transcript $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ comes from an execution between instances of \mathcal{S} and the adversary, where at least at one flow is not **Compute**-oracle-generated, it means that $(A^*, (g^{s_i})_i)$ has been simulated and B^* has been generated by the adversary. Since the server accepted a non-**Compute**-oracle-generated, it means that the biometric corrupt query has been made for the corresponding client \mathcal{C} . Thereafter, the same analysis, according to the secret key-corrupt status and the password-corrupt status, as above can be done.

We can thus conclude with

$$\Pr[\text{AskH}_4] \leq \sum_{\mathcal{C}} \mathcal{D}(q_{\mathcal{C}}) + \frac{q_h}{q} + q_h \cdot \text{Succ}_g^{\text{cdh}} (T + 4\tau_e).$$

□

5 Discussion

5.1 Optimality and Tightness

The authentication probability upper bound presented in Theorem [1](#) has two leading terms which are

$$q_S \left(\varepsilon_{\text{fa}} + \frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N} (t-1)!} \right) \quad \text{and} \quad \sum_{\mathcal{C}} \mathcal{D}(q_{\mathcal{C}}).$$

If ℓ is large enough, then the last two terms in the parenthesis are negligible, that is why we focus on the two terms which cannot be made negligible even with larger parameters: $q_S \cdot \varepsilon_{\text{fa}}$ and $\sum_{\mathcal{C}} \mathcal{D}(q_{\mathcal{C}})$. We claim that our scheme is optimal

and the security result is tight: these two terms could not be avoided, with any better protocol.

Let us consider the following adversary: \mathcal{A} asks for both a password and a secret key corrupt queries and then tries to authenticate using her own biometric. Every time she tries to authenticate, her success probability is equal to the false acceptance probability. Thus, her global success probability is approximately equal to $q_S \cdot \varepsilon_{fa}$. This attack is generic, independent of any specific protocol, and therefore this shows that the first upper bound cannot be avoided by any cryptographic means.

Secondly, let us consider the adversary which asks for all the secret key-corrupt and (liveness assumption) biometric-corrupt queries, against all the clients: the system is now protected by the passwords only. Thereafter, for each client \mathcal{C} , she makes q_C impersonation attempts with the server, using the q_C most probable passwords. For every client \mathcal{C} , the success probability is upper-bounded by $\mathcal{D}(q_C)$, therefore the global success probability is approximately equal to $\sum_{\mathcal{C}} \mathcal{D}(q_C)$ (it shows that the best attack consists in trying the most probable passwords against as many clients as possible). Once again, the adversary is generic and independent of any protocol. Therefore, this bound cannot be avoided either.

The other terms being negligible, our global upper-bound against authentication is tight, and our protocol optimal. The same way, one can show optimality and tightness for the semantic security.

5.2 Practical Parameters

Let us see what it gives with practical values. An iris scan is usually encoded over $N = 1024$ bits and $t = 300$ is considered as a good threshold for the Hamming distance between two measurements of the same biometrics. With such parameters, the false acceptance rate is estimated to 2^{-14} . For a similar false rejection rate, we can assume $\tau = 400$ as a reasonable threshold. In this case, if $\ell \geq 4$ then

$$\frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N} (t-1)!} \leq \frac{\binom{400}{100}}{2^{400}} + \frac{2^{3000}}{2^{2896} (299)!} \leq \frac{2^{321}}{2^{400}} + \frac{2^{104}}{2^{2033}} \leq 2^{-78}.$$

Note that ℓ is the length of the authentication tags. The shorter they are, the more efficient the protocol is, from a communication point of view. Can we reduce this value ℓ ? Consider an adversary that has corrupted both the password and the secret key. With very high probability (greater than ε_{fa}) the Hamming distance between a measurement of the adversary biometric and a client reference biometric is approximately 512 and so there are 512 indices i such that $\alpha_i = \alpha'_i$. If $\ell = 1$ there are approximately 512/2 other α_i and α'_i which are equal, that is, there are 768 indices i for which $\alpha_i = \alpha'_i$ and the adversary is able to impersonate the client. Therefore, if $\ell = 1$, with probability greater than the false acceptance probability an adversary can authenticate.

This means that ℓ must be greater than 2, and the previous bound shows that $\ell = 4$ is a good choice. However if one wants to guarantee the correctness of an

honest execution (for all the indices i , $\alpha_i = \alpha'_i$ and $\beta_i = \beta'_i$ if and only if the biometric bits are the same), then a good solution is to choose a greater ℓ . If $\ell = 24$, an honest execution succeeds with probability $2^{-14} \approx \varepsilon_{\text{fa}}$.

Another solution to guarantee the correctness of an honest session is to add a distillation step [10] after the protocol. Distillation allows two entities, with two secret keys with small Hamming distance, to agree on a common secret key, at the price of revealing some of the bits of the original secret keys. With a distillation step, one can choose $\ell = 4$. Even if the resulting secret is shorter than the original ones, this is not a problem in our case, since the original ones are quite large. The distillation step also allows to prevent some denial-of-service attacks where the adversary flips some of the α'_i (this is possible only if the liveness assumption is broken) or β_i . If for this i , $K_i = K'_i$ then with high probability the two entities will generate two different secret keys, whereas they both accept (a few modifications might not flip the decision), and then think that they share the same secret key. With a classical key confirmation, this attack can be detected, and the affected sessions identified. However the advantage of the distillation is that it allows to correct the errors introduced by the adversary or due to hazard, and then to avoid replaying the protocol once more.

5.3 Conclusion

In this paper, we defined a quite strong security model, since it allows a lot of information leakage for the adversary. It guarantees that the adversary has to break all the protections to impersonate a client. Namely, as long as the secret key is not recovered from the secure device, one can show that the success probability of the adversary against our scheme is negligible. As the unclonable device is probably the strongest and the most realistic protection, we can say that our protocol is quite secure.

Acknowledgment

This work has been partially supported by the French RNRT/ANR BACH Project, and the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

1. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
2. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
3. Barral, C., Coron, J.-S., Naccache, D.: Externalized fingerprint matching. In: Zhang, D., Jain, A.K. (eds.) ICBA 2004. LNCS, vol. 3072, pp. 309–315. Springer, Heidelberg (2004)

4. Bellare, M., Canetti, R., Krawczyk, H.: Modular approach to the design and analysis of key exchange protocols. In: ACM STOC Annual ACM Symposium on Theory of Computing, pp. 419–428. ACM Press, New York (1998)
5. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS Annual Symposium on Foundations of Computer Science, pp. 394–403. IEEE Computer Society Press, Los Alamitos (1997)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
8. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
9. Bellare, M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (May 1992)
10. Bennett, Brassard, Crepeau, Maurer: Generalized privacy amplification. In: ISIT (1994)
11. Bhargav-Spantzel, A., Squicciarini, A.C., Modi, S., Young, M., Bertino, E., Elliot, S.J.: Privacy preserving multi-factor authentication with biometrics. In: Juels, A., Winslett, M., Goto, A. (eds.) Proceedings of ACM DIM 2006 workshop, pp. 63–72. ACM Press, New York (2006)
12. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 82–91. ACM Press, New York (2004)
13. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)
14. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
15. Brainard, J.G., Juels, A., Rivest, R.L., Szydlo, M., Yung, M.: Fourth-factor authentication: somebody you know. In: ACM Conference on Computer and Communications Security, pp. 168–178 (2006)
16. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
17. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
18. Canetti, R., Krawczyk, H.: Security analysis of IKE’s signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002), <http://eprint.iacr.org/2002/120/>
19. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
20. Daugman, J.: How iris recognition works. In: ICIP (1), pp. 33–36 (2002)
21. Daugman, J.: Iris recognition and anti-spoofing countermeasures. In: 7th International Biometrics Conference (2004)

22. Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Transactions on Information Theory* 22(6), 644–654 (1976)
23. Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust fuzzy extractors and authenticated key agreement from close secrets. In: Dwork, C. (ed.) *CRYPTO 2006*. LNCS, vol. 4117, pp. 232–250. Springer, Heidelberg (2006)
24. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
25. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
26. Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences* 28(2), 270–299 (1984)
27. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC 1985)*, pp. 291–304 (1985)
28. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing* 17(2), 281–308 (1988)
29. Jakobsson, M., Myers, S.: *Phishing and Counter-Measures*. John Wiley and Sons Inc., Chichester (2006)
30. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: *ACM CCS 1999*, November 1999, pp. 28–36. ACM Press, New York (1999)
31. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
32. Valencia, V.: *Biometric Liveness Testing*, ch. 8. Osborne McGraw-Hill, New York (2002)

Repelling Detour Attack Against Onions with Re-encryption^{*}

Marek Klonowski, Mirosław Kutylowski, and Anna Lauks

Institute of Mathematics and Computer Science, Wrocław University of Technology
Marek.Klonowski@pwr.wroc.pl, Mirosław.Kutylowski@pwr.wroc.pl,
Anna.Lauks@pwr.wroc.pl

Abstract. This paper is devoted to ModOnions – an anonymous communication protocol, for which a message is encoded as a set of *onions* and sent through intermediate nodes so that each node knows only its predecessor and its successor on the routing path. Moreover, encoding details enable universal re-encryption: each node re-encrypts the message so that no observer can link together the ciphertexts before and after re-encryption and re-encryption can be performed without any public key. ModOnions were supposed to offer many additional features over classical onion protocols, such as resilience against replay attack. However, during ISC'2006 George Danezis presented a *detour attack* against this construction. It enables to redefine the routing path by inserting intermediate corrupt nodes between each two nodes of the original routing path. In this way anonymity becomes completely broken. We show that after slight changes in the protocol the attack does not work anymore. The patch proposed can also be seen as a general method of enforcing who is the final addressee of a message encrypted with the ElGamal scheme and multiple public keys.

1 Introduction

Onion-Routing is one of the most important techniques supporting anonymous communication in distributed systems. It is based on the following idea: messages are not sent from the sender to the receiver directly, but through a path of intermediate nodes chosen at random. Each message is encoded in a special way before being sent – i.e. it is encapsulated many times with asymmetric encryption. The resulting structure is called *an onion*. At each intermediate node on the path, one encryption layer is removed. Due to encryption properties, two messages entering and leaving the same node become indistinguishable from the point of view of an external observer. Such a situation is called *a conflict*. Intuitively, when many conflicts occur, then the batch of all messages becomes more and more mixed and linking the senders and the receivers becomes infeasible for an external adversary. Formal probabilistic analysis of this phenomenon in various models can be found for example in [1,6,9].

^{*} Partially supported by Polish Ministry of Science and Higher Education, grant N N206 2701 33.

Paper [5] presents a routing scheme with “onion-like” encoding that was supposed to protect users not only from a plain traffic analysis, but also from some classes of active attacks - in particular against so called *replay attack*. Such an attack is based on the following idea: the adversary controls some nodes in the network. Let us assume that he would like to trace the route of a particular message. He sends this message twice and traces repetitions among the messages transferred on all servers under his control. Finding such a repetition definitely betrays the route traced. The scheme from [5], called ModOnions, was built on the top of a very useful cryptographic primitive - an extended version of ElGamal encryption scheme with universal re-encryption (or URE, for short) introduced by Golle et al. in [4].

In [2] George Danezis presented an attack on ModOnions that can be successfully carried out by an active adversary that collaborates with even a small subset of nodes in the network. To some extent, this attack can be also applied to scheme presented in [8]. The idea is that the adversary manages to detour the routing path of a message. Namely, he or she modifies the encoded message in such a way that a corrupted node (i.e. controlled by the adversary) is inserted between each two subsequent nodes from the path. The adversary may insert markers that make it easy to recognize the messages from the same path. In this way the complete routing path may be revealed by the adversary and anonymity protocol becomes completely broken.

Main Result. In this paper we present a simple patch for ModOnions that protects the scheme against the attack of George Danezis. The patch proposed can also be seen as a general method of enforcing who is the final addressee of a message encrypted with the ElGamal scheme and multiple public keys, where each participant partially decrypts the ciphertext concerned. The method is based on different pairs of keys for the final decryption and for a partial decryption.

2 ModOnions – Onion Routing with URE

First, let us recall an extension of the ElGamal encryption scheme with universal re-encryption feature [4]. This scheme serves as the main building block for further considerations. It can be regarded as a simple, yet powerful extension of ElGamal encryption scheme that allows re-encryption of any ciphertext without knowledge of the public key.

Universal Re-Encryption. Let us recall ElGamal encryption scheme: we assume that p is a prime number such that the discrete logarithm problem is hard in \mathbb{Z}_p^* and g is a generator of \mathbb{Z}_p^* . Then a private key is a non-zero $x < p-1$ chosen uniformly at random, the corresponding public key is y , where $y := g^x \bmod p$. For a message $m < p$, a ciphertext of m is a pair (s, r) , where $r := g^k \bmod p$ and $s := m \cdot y^k \bmod p$ and $0 < k < p-1$ is chosen at random.

The ElGamal scheme is a probabilistic one: the same message encrypted for the second time yields a different ciphertext with overwhelming probability. Moreover, given two ciphertexts, it seems to be infeasible in practice to say whether they have been encrypted under the same key (unless, of course, the decryption key is given). This property is called *key-privacy* (see [4]). ElGamal cryptosystem has yet another important property. Everyone can re-encrypt a ciphertext (α, β) and get (α', β') where $\alpha' := \alpha \cdot y^{k'}$ mod p , $\beta' := \beta \cdot g^{k'}$ mod p for $k' < p$ chosen at random and public key y . Moreover, without the decryption key it is infeasible to find that (α, β) and (α', β') correspond to the same plaintext.

In [4] Golle et al. proposed a slightly modified version of this scheme that allows to perform re-encryption without the public key. The method proposed is also called universal re-encryption scheme or *URE* for short. It consists of the following procedures:

Setup: A generator g of a cyclic group G is chosen, where discrete logarithm problem is computationally hard. Then G and g are published.

Key generation: Alice chooses a private key x at random; then the corresponding public key y is computed as $y = g^x$.

Encryption: To encrypt a message m for Alice, Bob generates uniformly at random values k_0 and k_1 ($k_0, k_1 < p$). Then, the ciphertext of m is a quadruple: $(\alpha_0, \beta_0; \alpha_1, \beta_1) := (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})$. Let us note that this is a pair of two ElGamal ciphertexts with plaintext messages m and 1 (neutral element of G), respectively.

Decryption: Alice computes $m_0 := \frac{\alpha_0}{\beta_0^x}$ and $m_1 := \frac{\alpha_1}{\beta_1^x}$. Message m_0 is accepted if and only if $m_1 = 1$.

Re-encryption: Two random values k'_0 and k'_1 are chosen. Then we compute: $(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1})$, which is a re-encrypted version of the same ciphertext.

Throughout this paper $E_x(m)$ denotes an URE ciphertext of a message m for a secret decryption key x . Note that there are many possible values for $E_x(m)$, since URE is a probabilistic encryption scheme.

Extension of Universal Re-Encryption. Let us assume there are λ distinct servers on each routing path. We would like to encrypt a message in such a way that it must be processed by all servers on the path to obtain the plaintext. On the other hand, we would like to retain universal re-encryption feature to enable strong anonymization at each server.

Key setup: A private key x_i is chosen uniformly at random for the i th server ($1 \leq i \leq \lambda$). Then the corresponding public keys $y_i := g^{x_i}$ are computed and published.

Encryption: To encrypt a message m , values k_0 and k_1 are generated at random. The ciphertext has the following form:

$$E_{x_1, x_2, \dots, x_\lambda}(m) = (\alpha_0, \beta_0; \alpha_1, \beta_1) := (m \cdot (y_1 y_2 \dots y_\lambda)^{k_0}, g^{k_0}; (y_1 y_2 \dots y_\lambda)^{k_1}, g^{k_1}).$$

Obviously, $y_1 y_2 \dots y_\lambda$ is a kind of *cumulative* public key, since

$$E_{x_1, x_2, \dots, x_\lambda}(m) = \left(m \cdot g^{k_0 \sum_{i=1}^\lambda x_i}, g^{k_0}; g^{k_1 \sum_{i=1}^\lambda x_i}, g^{k_1} \right).$$

Moreover, $E_{x_1, \dots, x_\lambda}(m)$ is a ciphertext of m with the decryption key equal to $\sum_{i=1}^\lambda x_i$. Hence it can be re-encrypted in a regular way. Moreover, such a ciphertext can be *partially decrypted*, for instance, by the first server. Namely, it computes $E_{x_2, \dots, x_\lambda}(m)$ as the following quadruple:

$$(\alpha'_0, \beta'_0; \alpha'_1, \beta'_1) := \left(\frac{\alpha_0}{\beta_0^{x_1}}, \beta_0; \frac{\alpha_1}{\beta_1^{x_1}}, \beta_1 \right).$$

It is obvious that it still is a correct URE ciphertext for the “reduced” key $\sum_{i=2}^\lambda x_i$, and therefore it also can be re-encrypted as it was described above.

3 ModOnions Protocol

In this section we recall ModOnions – routing scheme from [5] based on URE encryption. Let x_i denote the secret key of server i . We assume that the corresponding public key $y_i = g^{x_i}$ is globally known. Again, g it is a generator of a group such that discrete logarithm is computationally hard. We assume that g is a public parameter shared by all servers.

Construction of an ModOnion. Let us assume that a server would like to send a message to a server s_λ . As the first step, a path of intermediate servers $s_1, s_2, \dots, s_\lambda$ is chosen at random. Then a modified onion (ModOnion) is built as a collection of λ ciphertexts, called *blocks*. The i th block, for $1 \leq i \leq \lambda - 1$, has the form: $E_{x_{s_1} + \dots + x_{s_i}}$ (send to s_{i+1}). The last block has the form: $E_{x_{s_1} + \dots + x_{s_\lambda}}(m)$. Let us note that this construction deviates from the encapsulation idea that is the core feature of the regular onions (see for instance [9, 6]).

Routing a ModOnion. First, all blocks described above are permuted at random and sent together to server s_1 . When a server s_i receives a ModOnion, it partially decrypts and re-encrypts all its blocks:

Partial decryption: each block $(\alpha_0, \beta_0; \alpha_1, \beta_1)$ is replaced by

$$\left(\frac{\alpha_0}{(\beta_0)^{x_i}}, \beta_0; \frac{\alpha_1}{(\beta_1)^{x_i}}, \beta_1 \right).$$

After this phase, exactly one block should contain a plaintext with the name of the next server on the path (i.e. s_{i+1}). This block is replaced with a random string pretending to be a URE ciphertext.

Re-encryption: All blocks are independently re-encrypted as described in Subsection 2.

Permuting: The blocks are permuted at random.

Sending to the next destination: The resulting ModOnion is sent to the next server on the path, i.e. to s_{i+1} .

As remarked in [5], such a scheme is vulnerable to multiplicative attack: one can multiply a message encrypted with an URE scheme by an arbitrary value γ . Indeed, $(\gamma \cdot \alpha_0, \beta_0; \alpha_1, \beta_1) = ((\gamma \cdot m) \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})$. The adversary processing a ModOnion can multiply an arbitrary block by γ . Then, if the adversary controls a significant number of nodes, it is pretty likely that one of nodes under his control gets a message of the form $\gamma \cdot s$, where s is a correct name of a server. In such a situation the adversary can be sure that this was the previously marked message.

To avoid these attacks an *investigation procedure* is executed, when any misbehavior is detected. If an honest node receives an invalid ModOnion - i.e. none of blocks or more than one decrypted block represents a name of the next server on the route or a valid final message, it complains against the previous server from the path. Then both servers - the previous server as well as the complaining one - prove that they behaved correctly. If one of them fails to prove its correct behavior, it is recognized guilty. If they manage to prove their correct behavior, the next predecessor on the path is interrogated. The procedure is repeated until a cheater is detected. Servers prove their correct behavior in a standard way using zero knowledge proofs for discrete logarithms (more details can be found in [5]).

4 Detour Attack on ModOnions

Preliminaries. We recall the attack of George Danezis [2] on the ModOnion scheme. It utilizes two observations:

1. Given $E_x(m)$, one can easily add an arbitrary value x' to the private key and create $E_{x+x'}(m)$. Indeed, if
$$E_x(m) = (\alpha_0, \beta_0; \alpha_1, \beta_1) = (m \cdot (y)^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}) ,$$

then one can get $E_{x+x'}(m)$ as the following quadruple:

$$(\alpha_0 \cdot (\beta_0)^{x'}, \beta_0; \alpha_1 \cdot (\beta_1)^{x'}, \beta_1) = (m \cdot (y \cdot y')^{k_0}, g^{k_0}; (y \cdot y')^{k_1}, g^{k_1}) .$$

In context of the ModOnions scheme it means that one can add an additional cryptographic layer to an arbitrary block.

2. Given $E_x(m) = (\alpha_0, \beta_0; \alpha_1, \beta_1)$, everyone can create a ciphertext of an arbitrary message m' for the same (unknown) secret key. Indeed, it suffices to compute $(m' \cdot \alpha_1, \beta_1, \alpha_1, \beta_1)$ and re-encrypt it.

Attack Description. The aim of an adversary is to find the destination of a message passing through a server controlled by the adversary. The attack is performed in consecutive steps, at each step the adversary reveals one subsequent node from the routing path of the message traced. Let us assume that an ModOnion:

$$E_{x_{s_i}}(s_{i+1}), E_{x_{s_i}+x_{s_{i+1}}}(s_{i+2}), E_{x_{s_i}+x_{s_{i+1}}+x_{s_{i+2}}}(s_{i+3}), \dots, E_{x_{s_i}+x_{s_{i+2}}+\dots+x_{s_\lambda}}(m)$$

arrives to server s_i controlled by the adversary (for simplicity of description we ignore here the random order of blocks at their arrival). After partial decryption ModOnion takes the form:

$$E_0(s_{i+1}), E_{x_{s_{i+1}}}(s_{i+2}), E_{x_{s_{i+1}}+x_{s_{i+2}}}(s_{i+3}), \dots, E_{x_{s_{i+1}}+x_{s_{i+2}}+\dots+x_{s_\lambda}}(m),$$

so the adversary knows that the ModOnion should be sent to s_{i+1} – the next server on the path. In his next move the adversary adds an additional layer with a random key x' to each block and he also replaces the decrypted block $E_0(s_{i+1})$ with a block $E_{x_{s_{i+1}}}(s_i)$. This block would redirect the ModOnion back to a node under control of the adversary in the next step. So now the ModOnion contains the following blocks:

$$E_{x_{s_{i+1}}}(s_i), E_{x_{s_{i+1}}+x'}(s_{i+2}), E_{x_{s_{i+1}}+x_{s_{i+2}}+x'}(s_{i+3}), \dots, E_{x_{s_{i+1}}+x_{s_{i+2}}+\dots+x_{s_\lambda}+x'}(m).$$

The adversary sends such a ModOnion from s_i to the node s_{i+1} . According to the rules of the protocol the node s_{i+1} removes one encryption layer – after the decryption phase performed by the node s_{i+1} the traced ModOnion consists of the following blocks:

$$E_0(s_i), E_{x'}(s_{i+2}), E_{x_{s_{i+2}}+x'}(s_{i+3}), \dots, E_{x_{s_{i+2}}+\dots+x_{s_\lambda}+x'}(m).$$

According to the protocol, server s_{i+1} sends them back the ModOnion to s_i . Of course, s_i can partially decrypt the blocks with the key x' and find s_{i+2} . Moreover, the decryption result is a correct ModOnion that would be obtained by s_{i+1} , if there were no attack. Exactly in the same way the adversary can find the subsequent servers s_{i+3}, s_{i+4}, \dots and the destination s_λ as well as the message m sent.

Of course, in order to avoid suspicious *pinging back* a ModOnion to the same node, the adversary may detour the ModOnion to any server under his control instead of s_i . For his convenience, he may also add an additional block containing a tag, which links the incoming ModOnion with the one, on which detour attack is exercised at s_i .

Tagging Attack. Another idea from [2] is to replace the random block by the block encoding the name of the next server. In order to check that the next t hops of the message are through nodes s_{j_1}, \dots, s_{j_t} , he inserts $E_{x_{j_1}+\dots+x_{j_t}}(\omega)$ instead of a random block, where ω is a special string used for this purpose. Now, if s_{j_t} becomes ω , then it is a strong evidence that the path went through exactly the nodes s_{j_1}, \dots, s_{j_t} and the ModOnion arriving with ω is the same as the ModOnion marked.

This attack is not particularly useful, if only one ModOnion is sent along the same path. However, it becomes dangerous, if ModOnions are used as a connection protocol - then one can try many times to discover the routing path.

5 Protocol Immune Against Detour Attack

In this section we present an improved version of the protocol recalled in Section 3. The core idea is as follows: each node has two pairs of keys. The first pair

of so called *transport keys* is used for transporting blocks through intermediate servers (like for the ModOnions protocol). The second pair of so called *destination keys* is used for encrypting and decrypting messages and routing addresses for their recipients. So there are the following keys of a server s :

- transport keys: a private key x_s and a public key $y_s := g^{x_s}$,
- destination keys: a private key x_s^* and a public key $y_s^* := g^{x_s^*}$.

Modified Construction of a ModOnion. Now the blocks encoding intermediate nodes on the routing path $s_1, s_2, \dots, s_\lambda$ are constructed as follows:

1. The block encoding the second node - s_2 has the form: $E_{x_{s_1}^*}$ (send to s_2).
2. For all $2 \leq i \leq \lambda - 1$, the block encoding address s_{i+1} has the form:

$$E_{x_{s_1} + \dots + x_{s_{i-1}} + x_{s_i}^*}(\text{send to } s_{i+1}).$$

3. The block encoding the payload message m has the form:

$$E_{x_{s_1} + \dots + x_{s_{\lambda-1}} + x_{s_\lambda}^*}(m, t),$$

where t is a code of the current time.

Note that each destination key x_i^* is used exactly once for each ModOnion, in the remaining cases for server i transport keys are used. For the simplicity of the security discussion we assume that all servers on the routing path are different.

Routing. Processing a Modified ModOnion by a server s_i becomes slightly more difficult:

1. Server s_i copies all blocks of a ModOnion. Then it decrypts all blocks with its private destination key. If every previous server on the path is honest, exactly one of the blocks after decrypting should contain a correct name of the next server s_{i+1} , the other blocks should yield meaningless strings.
2. If all of the blocks are meaningless strings, the investigation procedure is started. Otherwise all copies of the blocks (the original blocks obtained from the previous server), except for the one containing s_{i+1} , are decrypted with the private transport key of s_i . The blocks obtained are then re-encrypted in the regular way.
3. A random block replaces the block containing s_{i+1} .
4. The blocks are permuted at random.
5. The resulting ModOnion is sent to s_{i+1} .

The following steps are executed by the destination server:

1. It decrypts all blocks with its private destination key.
2. If all of the blocks are meaningless strings, then the investigation procedure starts.
3. If a correct message m, t has been already delivered before, then the investigation procedure starts.

5.1 Encoding Addresses and Countermeasures Against Multiplication Attack

We have to prohibit changing an address encoded in a block to the address of a chosen corrupt server s' . Note that this might be quite easy for the adversary: if he guesses that a block encodes an address s - i.e. it has the form $(\alpha_0, \beta_0; \alpha_1, \beta_1) = (s \cdot (y)^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})$, then the modified block encoding the address s' would be $(\alpha_0 \cdot s'/s, \beta_0; \alpha_1, \beta_1)$. The only danger for the adversary is that the guess was wrong and the block contained an address z is different from s . In this case the modified block would encode the address $z \cdot s'/s$, which can be an address of an honest server or can be invalid (i.e. it does not correspond to any node in the network). In the last case an investigation procedure may be started in order to determine the corrupt server. Probability that such an attack is detected (and an investigation procedure is started) increases, if it is unlikely that for any addresses z, s, z', s' the following equality holds for some α :

$$z \cdot \alpha = s \text{ and } z' \cdot \alpha = s'.$$

Potentially, there are many encoding schemes that may achieve this property. A simple way is to replace address z by $z|H(z)$, where $|$ denotes concatenation symbol and H is a strong hash function. Then we ask for pairs z, s, z', s' such that

$$\frac{z|H(z)}{s|H(s)} = \frac{z'|H(z')}{s'|H(s')}.$$

Finding such pairs seems to be infeasible for a good hashing function.

Another technique that can be applied on top of the previous method is the following modification of processing the blocks:

- During re-encryption of a ModOnion an additional operation is performed on each block $(\alpha_0, \beta_0; \alpha_1, \beta_1)$: the server tosses a coin and if tails are obtained, then it replaces this block by $(\alpha_0^2, \beta_0^2; \alpha_1^2, \beta_1^2)$. So if the original block encoded a message d , then it will encode d^2 .
- Constructing blocks by the sender becomes slightly more involved. In order to encrypt a string s it is necessary to compute $\bar{s} = \sqrt[\lambda]{s} \bmod p$. (This requires p to be chosen so that computing square roots is computationally easy.) Then inside a block that would contain s in the original design we encrypt \bar{s}^{2^i} , where i is chosen in the following way: If the block has to be read by the j th server on the routing path, then a symmetric coin is tossed $\lambda - j$ times and i is taken as the number of tails obtained during this experiment. This steps allow to hide the distance between the origin to the addressee of s , since the number of squarings executed is stochastically independent from this distance.
- Decryption of a ModOnion becomes slightly more difficult. After decryption with the private destination key, we obtain not s , but some root of the form $\sqrt[2^h]{s} \bmod p$, where $h \leq \lambda$ (according to our design, h oscillates around $\lambda/2$

regardless of the position on the routing path). In order to recognize s the server performs up to λ squarings on each decrypted block.

The main advantage of this technique is that even if an adversary correctly guesses the next address s_{i+1} on the routing path, it is uneasy to replace the address s_{i+1} by the address of a corrupt server z , as described before. Indeed, in order to remove s_{i+1} from block $(\alpha_0, \beta_0; \alpha_1, \beta_1)$ the adversary has to compute

$$(\alpha_0 / \bar{s}_{i+1}^{2^h} \cdot \bar{z}^{2^j}, \beta_0; \alpha_1, \beta_1)$$

where h is hidden in the block concerned. Note that h depends on independent decisions of the intermediate nodes and of the sender. The main trick here is that we insert *no* additional information bit to the intermediate addresses - according to the idea of ModOnion the only information got by a server on a routing path are the names of its successor and its predecessor. Nevertheless, we make hard to modify the next address, even if it is guessed correctly. In particular, this redundancy cannot be used for a replay attack.

6 Security of the Modified ModOnion Scheme

6.1 Immunity Against Detour Attack for Modified ModOnions

Using two different pairs of keys (transport and destination keys) is a simple way to make the ModOnion scheme immune against the detour attack. Indeed, in order to find the node s_{i+2} on the routing path a malicious node s_i should:

1. add a redirection block $E_{x_{s_{i+1}}^*}(s_i)$ in order to enforce the ModOnion to be sent back to him,
2. impose the additional encryption layer with a random key x' so that s_{i+1} gets only one plaintext after decrypting all blocks with its destination key.

Server s_{i+1} that is processing the ModOnion according to the protocol will decrypt the redirection block using his private destination key and remove one encryption layer from the remaining blocks using its transport key. So the layer encrypted with the destination key of s_{i+1} will not be removed from block that encodes the next address s_{i+2} on the original routing path. So the adversary will get no knowledge about s_{i+2} .

6.2 General Security Discussion

In this section we consider some security aspects of the proposed scheme.

Adversary Model: There are k servers in the system - s_1, s_2, \dots, s_k and the adversary can control a small fraction d of them (for example $d = 1/\lambda$). We also assume that the adversary can observe the inputs and the outputs of every server in the network. Finally, we assume that both the sender and the receiver are honest.

Attack Model: The adversary can observe the messages transmitted, manipulate messages processed by the servers controlled by him, and inject some new messages. In this paper we do not consider attacks based on traffic analysis that can be applied for regular onions as well.

We say that an attack \mathcal{A} succeeds if and only if the adversary can get some substantial information about the contents of a ModOnion and the probability that a corrupt server will be detected is negligible.

Offline Analysis. The first question to ask is if a single ModOnion may betray some knowledge about the information encoded inside it.

First assume that using algorithm \mathcal{A} the adversary gets some knowledge about the plaintext m encoded in a ModOnion. We show that in this case ElGamal encryption scheme would be insecure. Let (α, β) be an ElGamal ciphertext of some unknown plaintext created with public key y . Then we create a network and assign all key pairs except one destination key pair, where y is the public destination key. Then we create a ModOnion with appropriate blocks so that it points to the server with destination key y . We present such a ModOnion to \mathcal{A} and get some knowledge on m . This simple reduction is possible, since (α, β) does not depend on all other keys used for construction of the ModOnion.

The second case is that algorithm \mathcal{A} yields some knowledge about the identity of an intermediate node x on the routing path. If the adversary has some advantage, then there are values a, b such that x is limited to a and b and the following data are fixed: position of x on the routing path, the other nodes on the path and positions of the blocks corresponding to each node on the path. In this case \mathcal{A} distinguishes between ModOnions such that:

1. some blocks are created with transport key $y_a^{(t)}$ (and the keys of nodes different from x) and one block encodes a with destination key y and keys of nodes other than x ,
2. some blocks are created with transport key $y_b^{(t)}$ and one block encodes b with destination key y and keys of nodes other than x .

As we see, there is a relation between x occurring as a plaintext and transport key $y_x^{(t)}$ used for encoding information on other nodes on the path. Since we know the other plaintexts and their positions we can replace the plaintexts by 1's by dividing the first components of each such a block by the appropriate plaintext.

We show that we can use \mathcal{A} to distinguish between ElGamal ciphertexts. For this purpose, for a ciphertext (α, β) which has the form $(x \cdot y^k, g^k)$, where $x \in \{a, b\}$, we prepare inputs for \mathcal{A} . We construct two ModOnions: one created with $y_a^{(t)}$ assuming that (α, β) is a ciphertext of address a , and one created with $y_b^{(t)}$ assuming that (α, β) is a ciphertext of address b . One of these ModOnions is well formed and \mathcal{A} will tend to provide the correct answer for that input (note that thanks to re-encryption we may present many equivalent inputs). For the ill-formed ModOnion, behavior of \mathcal{A} might be arbitrary. If for ill-formed ModOnions with address a and key $y_b^{(t)}$, the answer is unbiased or biased to a ,

then we create ModOnions from (α, β) using l times $y_b^{(t)}$ (constant l depends on the bias). If there is a bias towards b , then (α, β) should encode b ; otherwise it should encode a .

Similarly, we consider the behavior of ill-formed ModOnions with address b and key $y_a^{(t)}$. The only uncovered case is that for ill-formed ModOnions with address b and key $y_a^{(t)}$ the answer tends to be a and for ill-formed ModOnions with address a and key $y_b^{(t)}$ the answer tends to be b . In this case we turn our attention to a block encoded with either $y_a^{(t)}$ or $y_b^{(t)}$: we expand it to a ModOnion, but in place of (α, β) we put a ciphertext of a . Then due to the properties of \mathcal{A} we may detect whether $y_a^{(t)}$ or $y_b^{(t)}$ was used.

Similarly we may show that the adversary gains no advantage if he considers a set of ModOnions from different time stages and tries to conclude if they come from the same routing path.

For the further discussion concerning active attacks notice that, if a ModOnion is partially decrypted with a key y , then a block, where (cumulative) public key Y was used, is transformed into a block corresponding to public key Y/y . So in particular if we use a wrong key, then the plaintext becomes unreadable until we reverse this operation and perform partial decryption corresponding to key y^{-1} , or there is nontrivial equality between two products of public keys used. Of course, the last case has negligible probability, if the keys are chosen at random.

Online Attacks. Using the nodes at his control, an active adversary may change the routing path of a ModOnion, manipulate the contents of a ModOnion without changing its route or inject additional ModOnions into the system.

So assume that ModOnion \mathcal{O} arrives at a corrupt server s . Then the adversary knows only the address of the next server on the route, say s_i . Assume that this node is not controlled by the adversary. At this moment the adversary may perform some actions listed above. Our goal is to show that either a manipulation will be detected with a non-negligible probability or with a high probability no additional information will be gained by the adversary.

First we argue that any change of the path of \mathcal{O} does not lead to a successful attack. The routing path can be changed in two different ways - such a path may use or not the blocks of \mathcal{O} . First assume that the adversary builds a path P of length l by himself so that:

1. l additional blocks that define P replace some original blocks of \mathcal{O} (if possible, they should replace random blocks),
2. the rest of original blocks are blinded with a secret key known to the adversary (in this way the adversary escapes the danger of an investigation that would be started after partial decryption and discovering two valid addresses),
3. the last server on the new route belongs to the adversary. Otherwise the ModOnion will be further processed, and since the chosen route is different from the original one with non-negligible probability, after some partial decryption no valid address or a final message will be found. This would lead to an investigation.

As the last server on the route belongs to the adversary, it can halt such ModOnions (the adversary can use the tags designed by G. Danezis in order to mark and detect these ModOnions). The key point is that even if P consists of servers from \mathcal{O} , the destination keys of the servers from this path will be used to decode the appropriate blocks added by the adversary. Since each destination key can be used only once per ModOnion no destination key will be used to decode the original blocks of \mathcal{O} . However, as we have mentioned a block partially decrypted with a wrong private key does not leak information on the plaintext from the blocks left from \mathcal{O} .

The second case is that the modified path of \mathcal{O} contains some links obtained from the blocks that encode the original path. As seen in Section 5.1, transforming an address encrypted in a block to another address is hard - even if we know the right block and the original address - we fail to get a valid address with a substantial probability. Similarly, using only some blocks encoding the links of the original ModOnion is risky, since we do not know their origin and destination, so the routing path may become disconnected. It is not disconnected, if we insert an additional subpath as a prefix and it terminates in s_i . However, this brings no advantage to the adversary, since no destination key will be used on this detour to decrypt the blocks from the original path.

It follows from the discussion above that the adversary should leave the route of \mathcal{O} unchanged. However, still the adversary may insert some data into random blocks and use them for revealing routing paths. We have to consider two cases:

Case 1. The inserted block is encoded with the cumulative keys that are not the same as for the original route. In that case the adversary is unable to distinguish between these blocks and random blocks.

Case 2. The inserted block is encoded by the transport keys of the servers from the path, say z_1, \dots, z_l . In that case the adversary observing that nodes can detect that \mathcal{O} was really processed by z_1, \dots, z_l . However, there are limitations of this method. First, the j th server on the route can insert at most j such tags into the ModOnion processed, since there are only j random blocks. (However, even this may be problematic. The adversary is sure only about the blocks, where it has detected a tag and the block where it puts a random block according to the original protocol.) Second, if the tag was well chosen (and it is detected by the addressee of the tag), there is no reason to resend it from the old point. If it is undetected, then the adversary must not try to put another tags and resend it. Indeed, in this case the same message would arrive twice at the destination point and an investigation would be started. The adversary has $\binom{k-k \cdot d}{l}$ possible paths of length l and $\gamma \leq \lambda - 1$ places for the tags. The probability that he guesses the right path (and inserts appropriate tags) equals $\frac{\gamma}{\binom{k-k \cdot d}{l}}$ assuming that the adversary knows the positions of all random blocks in the ModOnion. For example for $d = 1/\lambda$, the probability that he guesses one server from the original route is

$$\frac{\gamma}{k-k \cdot d} = \frac{\gamma}{k \cdot (1-1/\lambda)} < \frac{(\lambda-1) \cdot \lambda}{k \cdot (\lambda-1)} = \frac{\lambda}{k} .$$

One can also easily see that the chances of the adversary to detect a subpath are better, if the tags describe short subpaths. However, this requires that there are corrupt servers at the end of the subpaths, which cannot be guaranteed for short subpaths.

7 Concluding Remarks

We have proposed a modification of the ModOnions scheme that immunizes it against detour attacks introduced in [2]. It preserves main advantages of the original solution, which are minimal knowledge given to intermediate nodes on the routing path and use of universal re-encryption. The main disadvantage of the new scheme is increase of computational complexity.

References

1. Berman, R., Fiat, A., Ta-Shma, A.: Provable Unlinkability against Traffic Analysis. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 266–280. Springer, Heidelberg (2004)
2. Danezis, G.: Breaking Four Mix-Related Schemes Based on Universal Re-encryption. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 46–59. Springer, Heidelberg (2006)
3. Golle, P.: Reputable Mix Networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 51–62. Springer, Heidelberg (2005)
4. Golle, P., Jakobsson, M., Juels, A., Syverson, P.F.: Universal Re-encryption for Mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
5. Gomułkiewicz, K.M., Kutylowski, M.: Onions Based on Universal Re-encryption – Anonymous Communication Immune Against Repetitive Attack. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 400–410. Springer, Heidelberg (2005)
6. Gomułkiewicz, M., Klonowski, M., Kutylowski, M.: Provable Unlinkability Against Traffic Analysis Already After $\mathcal{O}(\log(n))$ Steps! In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 354–366. Springer, Heidelberg (2004)
7. Klonowski, M., Kutylowski, M., Lauks, A., Zagórski, F.: Universal Re-encryption of Signatures and Controlling Anonymous Information Flow. In: WARTACRYPT 2004 Conference on Cryptology, vol. 33, pp. 179–188. Tatra Mountains Mathematical Publications (2006)
8. Kutylowski, M., Klonowski, M., Zagórski, F.: Anonymous Communication with On-line and Off-line Onion Encoding. In: Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O. (eds.) SOFSEM 2005. LNCS, vol. 3381, pp. 229–238. Springer, Heidelberg (2005)
9. Rackoff, C., Simon, D.R.: Cryptographic Defense Against Traffic Analysis. In: ACM Symposium on Theory of Computing, vol. 25, pp. 672–681 (1993)
10. Tsiounis, Y., Yung, M.: On the Security of ElGamal Based Encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (1998)

Analysis of EAP-GPSK Authentication Protocol

John C. Mitchell¹, Arnab Roy¹, Paul Rowe², and Andre Scedrov²

¹ Stanford University*

² University of Pennsylvania**

Abstract. The EAP-GPSK protocol is a lightweight, flexible authentication protocol relying on symmetric key cryptography. It is part of an ongoing IETF process to develop authentication methods for the EAP framework. We analyze the protocol and find three weaknesses: a repairable Denial-of-Service attack, an anomaly with the key derivation function used to create a short-term master session key, and a ciphersuite downgrading attack. We propose fixes to these anomalies, and use a finite-state verification tool to search for remaining problems after making these repairs. We then prove the fixed version correct using a protocol verification logic. We discussed the attacks and our suggested fixes with the authors of the specification document which has subsequently been modified to include our proposed changes.

1 Introduction

The Extensible Authentication Protocol (EAP) [1] is an authentication framework developed by the Internet Engineering Task Force (IETF) which runs on the data link layer and supports a variety of mechanisms for two entities to authenticate themselves to each other. EAP is not itself an authentication protocol; rather it provides a context in which the entities can negotiate an authentication method such as Generalized Pre-Shared Key (GPSK) [2]. EAP is currently deployed on Point-to-Point connections, IEEE 802 wired networks, wireless LAN networks and over the Internet. The GPSK authentication method is a lightweight protocol being developed by the IETF EAP Method Update (EMU) working group. It uses symmetric cryptography and relies on a pre-shared key, as suggested by its name, between a server and a peer. The protocol seeks to minimize the number of messages exchanged, and hence is particularly well-suited for use in handheld devices where memory and computational resources are a limitation. The protocol is designed to provide mutual authentication and key agreement between the server and the peer. In this paper, we report improvements in the protocol and report on a formal analysis of the GPSK method. The main goal of our analysis is to have a positive impact on the protocol during a critical stage of the development and standardization process. As such, our suggested improvements attempt to keep the protocol largely in tact.

* Mitchell and Roy are partially supported by the NSF grant to Stanford and Penn, “Collaborative Research: High Fidelity Methods for Security Protocols” and by the TRUST NSF Science and Technology Center.

** Partially supported by OSD/ONR CIP/SW URI projects through ONR Grants N00014-01-1-0795 and N00014-04-1-0725. Additional support from NSF Grants CNS-0429689 and CNS-0524059 and from ONR Grant N00014-07-1-1039.

In this paper, we use finite-state model checking to find errors, and Protocol Composition Logic [34] to prove correctness after errors have been found and repaired. The model checker we use is called $Mur\phi$ [5]. $Mur\phi$ has been successfully used in the past on a variety of protocols including Kerberos [6], SSL [7], and the 802.11i 4-Way Handshake [8]. As a model checker, $Mur\phi$ is well suited for finding flaws but is insufficient to prove the correctness of a protocol. So to compliment $Mur\phi$ we use Protocol Composition Logic (PCL) [9] as a proof tool. $Mur\phi$ was useful in detecting some of the problems with the protocol specification as we first encountered it, while PCL was useful for proving that the fixes we suggested, and which were subsequently adopted, are correct.

Our analysis uncovered three weaknesses with GPSK. The first is a repairable Denial-of-Service attack against the peer, in which the attacker forces the peer to exhaust its memory thereby blocking the protocol. This attack is virtually identical to one which was found on the 802.11i 4-Way Handshake [8]. The second weakness is due to a non-standard use of the key derivation function which is used to create session keys. Although the non-standard use does not provide an obvious attack we cannot exclude the existence of an attack. In addition, we indicate the difficulties which such non-standard usage creates when trying to prove the protocol's correctness. Finally, we identify a ciphersuite downgrading attack in which the attacker can force the peer to choose a weak hash function or encryption algorithm. If the ciphersuite is susceptible to a key-recovery attack then the attacker can learn the session keys and then eavesdrop on all subsequent communications.

In addition to discovering these weaknesses we also suggest ways to fix them and prove that our proposed fixes make the protocol secure. As indicated above, we discussed the weaknesses and our suggested fixes with the authors of the GPSK specification. In turn, the authors presented the issues to the EMU working group for open discussion. They recognized the problems and they have since incorporated our suggested fixes to the most recent protocol specification. Our interaction with the authors came at a time in which GPSK was mature enough to undergo a thorough analysis, and yet early enough in the standardization process that it was not widely implemented.

There have been many efforts to develop and use methods for proving security properties of network protocols. In recent years, most efforts have used the so-called *symbolic model*, also referred to as the *Dolev-Yao* model [10,11,12]. In the symbolic model, protocol execution and the possible actions of an attacker are characterized using a symbolic model of computation that allows nondeterminism but does not incorporate probability or computational complexity bounds. In addition to many model checking and bug-finding efforts, there have been some significant correctness proofs carried using the symbolic model, including mechanically checked formal proofs [13,14], unformalized but mathematical proofs about a multiset rewriting model [15,16,17], and work using compositional formal logic approaches [18,19,20,21,22]. Several groups of researchers have taken steps to connect the symbolic model to the probabilistic polynomial-time *computational model* used in cryptographic studies, e.g., [23,24,25,26,27,28,29,34,30]. Protocol Composition Logic has been used to prove correctness of versions of Kerberos in the symbolic model [31], and in the computational model [32], with errors in the Diffie-Hellman variant of Kerberos and proofs of security

presented in [33]. Connections between symbolic trace properties and computational soundness properties are developed in [34]. All these efforts have been aimed at proving security properties for well-established protocols. Unfortunately, protocols such as EAP-GPSK are still being designed with flaws and weaknesses that are identical to those found and fixed in previous protocols. The major contribution of this work is to integrate the methods of protocol analysis into the standardization process before the protocol is deployed in a variety of implementations.

The rest of the paper is structured as follows. Section 2 describes in more detail the EAP framework and the GPSK method. In Section 3 we present the weaknesses we found and our suggested fixes. Section 4 describes the role that Mur φ played in our analysis. In Section 5 we present a proof of correctness of the fixed protocol. We conclude in Section 6.

2 EAP

The Extensible Authentication Protocol (EAP) [1] is an authentication framework which is meant to support a variety of authentication methods. No single authentication protocol is defined. Instead, it defines message types that allow an authenticator and a peer to choose and perform an authentication mechanism. EAP is designed to run on the data link layer where IP connectivity may not be available. It provides support for duplicate elimination and retransmission but relies on lower layers to properly order packets. The authenticator may act as a pass-through and use a backend authentication server. This allows an implementation in which only the end server must be configured for a particular method. The authenticator does not have to be updated with the introduction of every new authentication method. The distinction between the authenticator and the authentication server does not arise in our analysis of the GPSK authentication method. Therefore we will treat the authenticator and the server as one entity referred to simply as the server.

EAP was designed to work with Point-to-Point connections, and was subsequently adapted for IEEE 802 wired networks as well as wireless LAN networks and over the Internet. In each of these settings an attacker may be able to control the network. EAP assumes an attacker can perform such actions as eavesdropping network traffic, modifying or spoofing packets, and offline dictionary attacks among others. This allows an attacker to attempt such attacks as person-in-the-middle attacks, ciphersuite downgrading attacks, and key recovery attacks due to weak key derivation.

An EAP conversation typically consists of three phases: discovery, authentication and secure association. In the discovery phase the two agents must identify each other and negotiate an authentication method. Then they carry out the chosen method in the authentication phase. The secure association phase occurs when the authenticator and the authentication server are distinct entities and so does not affect the present analysis. Our focus in the current analysis is on the authentication phase.

2.1 EAP-GPSK

The Generalized Pre-Shared Key (GPSK) protocol [2] is an EAP authentication method which is meant to be lightweight and flexible. To this end, it uses symmetric

cryptography relying on a long-term, pre-shared key (denoted by PSK) between a server and a peer. The use of symmetric cryptography minimizes the computational resources required at either end of the communication, making it suitable for smartcards, handheld devices, or any device in which computational resources and memory are a significant constraint. To increase efficiency, the protocol also attempts to minimize the number of round trips. For flexibility the protocol allows for the negotiation of cryptographic ciphersuites which detail the encryption algorithm (if any), the message integrity mechanism and the key derivation algorithm the protocol participants will use. In this way, a server may authenticate several peers with a variety of local preferences for ciphersuites which may depend on the peers' computational constraints.

We show a successful message exchange in Fig. 1 at an abstract level. In the figure, S represents the server's ID and P represents the peer's ID. SNonce and PNonce are the server and peer nonces, respectively. CSuiteList represents the list of ciphersuites supported by the server, while CSuiteSel represents the ciphersuite selected by the peer. The field $\{\text{Payload}\}_{PK}$ represents an optional encrypted payload block that is a generic mechanism for exchanging confidential data. Higher level protocols may piggy-back on GPSK using the encrypted payload block to guarantee confidentiality. This means that sensitive confidential data may be sent as early as *Message 2*. *Messages 2, 3 and 4* each have a keyed message authentication code, MAC_{SK} , appended to the end. This is essentially a keyed hash of the rest of the message, although implementations depend upon the ciphersuite which is chosen.

[*Message 1: S* → *P*]: SNonce, S, CSuiteList
 [*Message 2: P* → *S*]: P, S, PNonce, SNonce, CSuiteList, CSuiteSel, $\{\text{Payload}\}_{PK}$, MAC_{SK}
 [*Message 3: S* → *P*]: PNonce, SNonce, CSuiteSel, $\{\text{Payload}\}_{PK}$, MAC_{SK}
 [*Message 4: P* → *S*]: $\{\text{Payload}\}_{PK}$, MAC_{SK}

Fig. 1. Successful GPSK message exchange

The keys SK and PK are both derived from a key derivation function KDF-X by way of an intermediate master key MK . KDF-X takes two arguments, a key and a seed, and outputs a bit string of length X. The notation $KDF-X(Y,Z)[i..j]$ represents the i 'th through j 'th octets (8 bits) of the output of the KDF-X. The PSK has length PL, while the SK and PK have length KS which is a value specified by the ciphersuite. Key derivation is defined as follows:

inputString = PNonce || P || SNonce || S.
 $MK = KDF-KS(0x00, PL || PSK || CSuiteSel || \text{inputString})[0..KS-1]$.
 $SK = KDF-\{128+2*KS\}(MK, \text{inputString})[128..127+KS]$.
 $PK = KDF-\{128+2*KS\}(MK, \text{inputString})[128+KS..127+2*KS]$.

The first 128 octets of $KDF-\{128+2*KS\}(MK, \text{inputString})[128+KS..127+2*KS]$ are divided into two keys which are exported as part of the protocol. They may be used for key derivation in higher level protocols. Every EAP method which supports key derivation is required to export such keys, but we omit them because they are not relevant to the current analysis.

GPSK is intended to provide mutual authentication between the peer and the server. After a successful message exchange the server should believe the peer is authentic due to the use of the key SK (derived from the long term key PSK) for the MAC in *Message 2*. Likewise, the peer should believe the server to be authentic due to the use of SK for the MAC in *Message 3*. GPSK is also intended to provide session independence. Even if the master key MK is compromised, this should not help an attacker to compromise past or future sessions. As with any symmetric key authentication protocol, the secrecy of the long-term key PSK is crucial for all of the above properties to hold. Unlike some protocols, GPSK does not support fast re-keying because the number of round trips is already at a minimum.

The peer and the server must silently discard any message which is unexpected (*e.g.* receiving *Message 4* instead of *Message 2*), doesn't parse (*e.g.* the wrong nonce is returned), or whose MAC is invalid. The only exception is for *Message 2*. If the server receives an invalid MAC then it must respond with an EAP failure message. The peer must always be willing to accept *Message 1* from a server since there is no integrity protection.

3 Anomalies

In our analysis of EAP-GPSK we found a number of anomalies. The first is a potential Denial-of-Service attack against the peer that is reminiscent of a similar attack found on the 802.11i 4-Way Handshake. We also identified a possible problem with the way in which the master key MK is derived. Lastly, we found a potential ciphersuite downgrading attack. Let us consider these issues one by one.

3.1 Denial-of-Service Attack

There is a simple Denial-of-Service attack that is made possible by the fact that *Message 1* provides no integrity protection. The result of the attack is a discrepancy between the SKs held by the peer and the server. This causes the peer to be unable to validate the MAC in *Message 3*. The protocol is blocked and the server will timeout and de-authenticate the peer. Obvious inspection shows that *Messages 2, 3 and 4* all have integrity protection. This means that if an attacker tries to forge these messages the peer can simply discard them when the MAC does not validate properly. We do not consider it an attack to cause the peer to attempt to validate a large number of MACs.

For simplicity the attack is explained in a situation where the peer can only maintain one open conversation with a given server. A message exchange for a successful attack is shown in Fig. 2. Only the relevant portions of each message are shown. When the peer receives a legitimate *Message 1* from a server it computes the MAC key SK, and responds with *Message 2*. At this point the attacker can send a fake *Message 1* with a new nonce. Since the peer can only have one open conversation with the server, and since the peer must always accept *Message 1*, the peer will recalculate the MAC key to a different value SK' when it receives the fake *Message 1* with a new nonce. When receiving a valid *Message 3*, the peer can no longer verify the MAC because it was calculated with the key SK, and the peer is trying to validate it with the new key SK'. At this point, the protocol can no longer proceed.

[Message 1:S → P]: SNonce, S
[Message 2:P → S]: P, S, PNonce, SNonce, MAC_{SK}
[Message 1':Attacker → P]: SNonce', S
 (The peer chooses a new PNonce' and calculates SK' .)
[Message 3:S → P]: PNonce, SNonce, MAC_{SK}
 (The peer uses SK' to check MAC_{SK} and verification fails.)

Fig. 2. A Successful DoS attack on GPSK

If the peer can only hold one open conversation with a server the attack requires a single, well-timed message from the attacker. This may be prevented by allowing the peer to maintain several open conversations with a server. However, in this case an attacker can flood the network with *Message 1'* to exhaust the peer's allocated memory. The fact that GPSK is designed to work well in devices with limited resources makes this type of attack more feasible than it otherwise might be. The peer's memory resources are likely to be very restricted, allowing an attacker to fill the peer's memory with relatively little effort. The ability to flood the network with fake messages also reduces the attacker's reliance on good timing.

One might argue that the attack can be prevented by not allowing the peer to respond to a new *Message 1* when it is waiting for *Message 3*. However, this is not a viable option. This solution introduces the possibility that the protocol will be blocked, even in the absence of an attacker. It also enables another DoS attack which is even easier to execute as we explain below.

The principal concern, as was explained in [8], is that *Message 2* will not reach the server. Since EAP is frequently run on wireless LANs, packet loss is a legitimate concern. If the server does not receive *Message 2*, it will re-transmit *Message 1* after an appropriate timeout. The peer will discard re-transmissions because it is waiting for *Message 3*, and the server will never get a response. Alternatively, an attacker can cause the same problem by sending one fake *Message 1* to a peer before the server initiates a conversation. This will force the peer into a state in which it is waiting for *Message 3*, and any attempts by a server to authenticate the peer will be ignored.

This anomaly is virtually identical to a DoS attack found in the 802.11i 4-Way Handshake and presented in [8]. The solution we propose is therefore analogous to the solution which was proposed by the authors of [8] and ultimately adopted by the 802.11 working group. The memory exhaustion attack is possible because the peer is forced to maintain state for each *Message 1* it receives. The proposed solution allows the peer to maintain state for each server regardless of the number of times it receives *Message 1*. Since the number of servers associated with a given peer is likely to be small, this should drastically reduce the memory used by a peer.

The first time a peer receives *Message 1* from a server it will choose a fresh PNonce and remember it. If it receives another *Message 1* from the same server before completing the protocol then the peer will re-use PNonce. *Message 2* will remain as it is. The fact that *Message 3* has a MAC whose key depends on the PSK allows the peer to trust its contents. Instead of storing the SK used in *Message 2*, the peer will recalculate SK when it receives *Message 3*. This solution requires *Message 3* to contain enough information for the peer to re-compute SK. Currently, *Message 3* does not

contain the server's ID which is necessary to compute SK. One way to fix this is to leave the message format as it is, and change the key derivation. If SK no longer depends on the server's ID, then the peer can compute it at this stage. Another option is to add the server's ID to *Message 3* and leave the key derivation alone. This would also provide the peer with all the information it needs to re-compute SK. Ultimately, the latter solution was adopted by the EMU working group.

Let us see how this solution defends against the attack in Fig. 2. When the attacker sends a fake *Message 1*, the peer no longer updates its own nonce. The peer will compute a new SK' for *Message 2*' based on the fake SNonce'. It will store neither SNonce' nor the new SK'. Now when the peer receives a legitimate *Message 3*, it can verify that PNonce was correctly returned, and it can use all the information provided to re-compute SK and verify the dependence on PSK. This will convince the peer that the legitimate server sent *Message 3*.

While this solution does not introduce integrity to *Message 1*, it does prevent the peer from changing state in response to an unauthenticated message. By re-using PNonce an attacker can now cause the peer to produce many copies of *Message 2* with the same key by sending many forged copies of *Message 1* using the same nonce. If the encrypted payload changes every time then this gives the attacker access to many different encryptions under the same key. The peer should be aware of this and choose a ciphersuite which is strong enough to resist cryptanalysis under these conditions.

3.2 Non-standard Key Derivation

The second anomaly we found involved the derivation of the master key MK. Recall from above that MK is derived by:

$$\text{MK} = \text{KDF-KS}(0x00, \text{PL} \parallel \text{PSK} \parallel \text{CSuiteSel} \parallel \text{inputString})$$

The use of "0x00" as the key to the KDF is not standard. In TLS [35] for example the `master_secret`, which corresponds to MK, is derived from a KDF which uses the long-term shared key (`pre_master_secret`) as its key input:

$$\text{master_secret} = \text{PRF}(\text{pre_master_secret}, \text{"master secret"}, \text{ClientHello.Random} + \text{ServerHello.Random})$$

The derivation of MK in GPSK does not provide an obviously reliable way for an attacker to learn MK or the keys SK and PK. It is unwise, however, to deviate too much from accepted standard usage which has undergone thorough investigation in other protocols.

The current derivation poses problems on theoretical grounds as well. Cryptographic implementations are often accepted because they provide strong guarantees when one assumes that the implementations satisfy standard assumptions. For example, the key derivation function for TLS is transparently assumed to act as a pseudo-random function (PRF). Let us now examine what assumptions about KDF are necessary for the current implementation to provide strong keys.

The KDF is being used in two different capacities, one for the MK derivation and another for the SK and PK derivations. In the MK derivation it acts like a hash function or a randomness extractor whereas in the SK and PK derivations, it acts like a PRF.

The cryptographic security of SK and PK are defined assuming uniformly random execution of a key generation algorithm. Assuming, for the particular schemes in use for this protocol, that the keys are sampled uniformly from the key space of a given length, we would require SK and PK to be computationally indistinguishable from purely random numbers of the same length.

Working backwards from the SK and PK derivations, if we model the KDF as a PRF, we can ensure that SK and PK are computationally indistinguishable from random, as long as MK itself is also computationally indistinguishable from random. So the requirement reduces to ensuring that MK is pseudo-random. What should the KDF with key $0x00$ behave like in order to ensure this? Let us denote by $kdf_{00}(y)$ the function $KDF(0x00, y)$. We have the following alternatives:

1. kdf_{00} is a Random Oracle: Then MK would indeed be perfectly random by definition. However, this is a very strong assumption and although a theoretically useful model, it is unrealizable in practice and its usage is debatable [36].

2. KDF is a PRF: This is too weak. It is possible to construct perfectly valid PRFs which output a constant if the key is $0x00$. This does not violate pseudo-randomness because a PRF's output 'seems' to be random with a randomly chosen key. The probability of a key being all 0's is exponentially small in the security parameter and hence this is a very low probability event.

3. kdf_{00} is a pseudo-random generator (PRG): The trouble with this model is that a PRG's output is claimed to be pseudo-random only if the input seed is uniformly random and unknown. However, for this protocol, parts of the seed are known (the nonces, IDs, ...) and neither is it uniformly random (IDs and CSuiteSel have structure).

Thus the alternatives that are weaker than the random oracle model do not guarantee strong keys. Perhaps the simplest solution to the problem is to use PSK as the key when deriving MK. In that case, if we assume that KDF is a PRF, MK will be indistinguishable from a random number of the same length because PSK is random and unknown. This implies that both SK and PK will be indistinguishable from random.

In talking with the authors of the specification it seems that the reason this approach was not taken from the start was because different ciphersuites have different key lengths and PSK might not be the right length for some of them. The working group finally decided to require PSK to be long enough for all current (and many future) ciphersuites. Then PSK will be truncated to be the right length if it is too long for the chosen ciphersuite.

3.3 Ciphersuite Downgrading Attack

The last anomaly we found was a potential ciphersuite downgrading attack. Just as with the DoS attack, this also arises from the fact that the first message has no integrity protection. An attacker who controls the network can modify a legitimate message from a server. In this case the attacker can change CSuiteList to include only weak ciphersuites.

In particular, the attacker might force the peer to choose a ciphersuite which does not provide an encryption mechanism, or which provides an encryption algorithm that the attacker can break in real time. In this case, any data which is passed in the encrypted payload block in *Message 2* can be read by the attacker.

There is an additional concern if the weak ciphersuite is susceptible to a key-recovery attack. If the attacker is able in real time to recover the session keys based only on the knowledge of the nonces and the IDs of the peer and the server, then he will break authentication. The attacker could block *Message 2* and replace the peer's nonce with a freshly chosen nonce, and compute the corresponding key to create a valid MAC. In this way the attacker could impersonate both the peer and the server giving him full control over their communication.

To counteract such an attack the protocol designers have required that CSuiteList in *Message 1* must contain two specified ciphersuites. While this should ensure that some of the ciphersuites offered meet some minimum strength, it does not ensure that the peer chooses the strongest ciphersuite which it supports. Although the peer will likely support both of these ciphersuites it has the freedom to choose a weaker ciphersuite if it is offered. In addition, one of the required ciphersuites does not support encryption. Although this ciphersuite is among those required to be supported because it does not suffer from a key-recovery attack, an unwitting peer may choose this ciphersuite and still attempt to send confidential data in *Message 2*.

This attack seems unavoidable as long as *Message 1* continues to lack integrity protection. As long as the peer is allowed to send encrypted data in *Message 2* before the CSuiteList is authenticated, this data might be sent even after choosing a ciphersuite without support for encryption. However, the damage of this attack can be fully avoided as long as the peer is aware of the potential problem and chooses a strong ciphersuite. Also, if the peer wishes to send confidential data, then it must choose a ciphersuite which supports encryption, and it must wait until *Message 4* to send the data. After discussing the problem with the protocol authors, the specification was amended to contain warnings for the peer.

4 Model Checking the Protocol

Finite-state verification tools such as Mur ϕ have proven to be very useful in the analysis of security protocols. Mur ϕ was successfully used in [6] to verify small protocols such as the Needham-Schroeder public key protocol, the Kerberos protocol, and the TMN cellular telephone protocol. In [7] Mur ϕ was also successfully applied to the analysis of the SSL 3.0 handshake protocol using a “rational reconstruction” methodology which was adopted in [8] to analyze the 802.11i 4-Way Handshake.

Tools such as Mur ϕ , commonly called model checkers, verify specified properties of a nondeterministic system by explicitly enumerating all possible execution sequences and checking for states which violate the specified properties. Although security protocols are infinite systems, many flaws have been found in finite (and very small) approximations to them. While finite verification has been successful in finding bugs, failure to find a bug does not mean the protocol is secure. Certain simplifications must be made

when creating a model and these may eliminate crucial details. The restriction to a finite state space with only a small number of participants is also a crucial limitation.

The use of Mur ϕ for the current analysis served two main goals. First, it was used to help detect the flaws found in the original specification of GPSK [2]. Second, we used it to perform a preliminary search for new flaws which may have been introduced by the fixes we proposed. Since a run of Mur ϕ in which no flaws are found does not imply the security of the protocol the next section is dedicated to the describing the process of giving a formal proof of the security properties.

To use Mur ϕ to verify the protocol we must create a model of the protocol following the specification, add a model of an attacker, state the security properties we would like to check and then run Mur ϕ on the model. In this last step Mur ϕ searches through all of the possible traces of the protocol checking at each step if any of the security properties fail. If so then it will return the trace which ends in the violation. This allows the user to see what caused the problem.

In formulating a model of the protocol the honest participants do not stray from the specification. They act deterministically. We assume that every pair of servers and clients shares a long term PSK. Since PSK is never used as a key we will assume that the attacker cannot recover any PSK. The MAC key SK is simply modeled as a list containing PSK and the inputString. Again, since Mur ϕ is not well-suited for discovering low level cryptographic attacks we assume that SK will remain secret. Thus our model contains no mechanism for the attacker to learn SK. Since the encrypted payload block is not explicitly used as part of the authentication mechanism, we exclude this component of the messages. For simplicity we model a good ciphersuite list and a bad one by a 1 or a 0 respectively. We similarly model a good (strong) ciphersuite selection and a bad one. Since we cannot model the lower level details of the ciphersuites (*i.e.* the way they function on bitstrings) this good/bad distinction should be enough to detect a ciphersuite downgrading attack.

Despite the above restrictions on the attacker, he still can act nondeterministically in a variety of ways. The attacker can eavesdrop and remember any message sent on the network. Since there is no encryption used, the attacker can read the plaintext of any message and decompose it into its various parts. He can block any message, and he can generate and send any message which is made up of components from other messages. In particular, this means that an attacker can replay complete messages. Since SK is assumed to be secure, the attacker will remember and use the MACs he sees as indecomposable units.

Since PSK is assumed to remain secret the properties we are concerned with in our analysis are mutual authentication and consistency of the key SK. Namely, at the end of a session the peer and the server must each believe they are talking to each other, and they must share the same key SK. We also check to see if the protocol is blocked by the attacker as it is in the DoS attack. Finally, we have Mur ϕ print a trace if a ciphersuite downgrading attack occurs. Once these attacks were detected we added a feature which allows us to turn these attacks off. This allows us to more efficiently search for other attacks.

Although Mur ϕ only creates finite models, it does allow us to specify parameters which will determine the size of the model. In this way, we can write one model that can

be scaled up or down simply by switching the values of these parameters. In modeling GPSK we chose to vary the number of peers and the number of servers. We can control the number of nonces available, the number of sessions to be completed and the number of distinct actions the attacker may execute.

Mur ϕ proved useful in successfully detecting and verifying the existence of the DoS attack as well as the ciphersuite downgrading attack. We then ran Mur ϕ on the model with the suggested fixes. Since our proposed fix for the ciphersuite downgrading attack does not change the message exchange, we simply turn the attack off in order to assume that the peer chooses a strong ciphersuite. We successfully verified that no errors exist when there is just one peer and one server engaging in up to 10 sessions. In addition we verified correctness with one peer and two servers engaging in 3 sessions total as well as the situation with two peers and one server engaging in up to 3 sessions. In attempting to verify the case with 2 peers and 2 servers engaging in 2 sessions total we ran out of memory. While the verification didn't run to completion Mur ϕ also failed to find any flaws before exhausting memory. This may be taken as tentative evidence of a lack of an attack since Mur ϕ prints a violating trace as soon as a problem is found. The model checking of this protocol was very familiar the previous experience of model checking the 802.11i 4-Way Handshake [8], so we did not investigate it further.

5 Proof of Correctness

In this section, we present a formal correctness proof of EAP-GPSK using *Protocol Composition Logic (PCL)* [37][38][39][40][41][19][18]. In previous work, PCL has been proved sound for protocol runs that use any number of principals and sessions, over both symbolic models and (for a subset of the logic at present) over more traditional cryptographic assumptions [3].

5.1 Overview of Proof Method

We begin with a brief discussion of PCL relevant to the analysis of EAP-GPSK.

Modeling protocols. A protocol is defined by a set of roles, each specifying a sequence of actions to be executed by an honest agent. In PCL, protocol roles are represented using a simple “protocol programming language” based on *cords* [37]. The possible protocol actions include nonce generation, signatures and encryption, communication steps, and decryption and signature verification via pattern matching. Programs can also depend on input parameters (typically determined by context or the result of set-up operations) and provide output parameters to subsequent operations.

Protocol Logic and the Proof System. For a summary of the proof system and the proof of soundness of the axioms and the rules, we refer the reader to [9][38][19]. Most protocol proofs use formulas of the form $\theta[P]_X\phi$, which means that starting from a state where formula θ is true, after actions P are executed by the thread X , the formula ϕ is true in the resulting state. Formulas ϕ and ψ typically make assertions about temporal order of actions (useful for stating authentication) and/or the data accessible to various principals (useful for stating secrecy).

The proof system extends first-order logic with axioms and proof rules for protocol actions, temporal reasoning, knowledge, and a specialized form of invariance rule called the *honesty rule*. The honesty rule is essential for combining facts about one role with inferred actions of other roles, in the presence of attackers. Intuitively, if Alice receives a response from a message sent to Bob, the honesty rule captures Alice's ability to use properties of Bob's role to reason about how Bob generated his reply. In short, if Alice assumes that Bob is honest, she may use Bob's role to reason from this assumption.

5.2 Formal Description of EAP-GPSK in the PCL Programming Language

```

GPSK : Server  $\equiv$  [
  new  $SNon\hat{c}$ ;
  send  $SNon\hat{c}.\hat{S}.CSL$ ;

  receive  $\hat{P}.\hat{S}.PNon\hat{c}.SNon\hat{c}$ .
     $CSL.CSS.enc1.mac1$ ;
   $MK := \text{prg } PSK$ ;
   $InputString := PNon\hat{c}.\hat{P}.SNon\hat{c}.\hat{S}$ ;
   $SK := \text{kdf1 } InputString, MK$ ;
   $PK := \text{kdf2 } InputString, MK$ ;
   $pl1 := \text{symdec } enc1, PK$ ;
   $\text{verifymac } mac1, \hat{P}.\hat{S}.PNon\hat{c}.SNon\hat{c}$ .
     $CSL.CSS.enc1, SK$ ;
   $enc2 := \text{symenc } pl2, PK$ ;
   $mac2 := \text{mac } PNon\hat{c}.SNon\hat{c}$ .
     $CSL.enc2, SK$ ;
  send  $PNon\hat{c}.SNon\hat{c}.\hat{S}.CSL.enc2.mac2$ ;

  receive  $enc3.mac3$ ;
   $\text{verifymac } mac3, enc3, SK$ ;
   $pl3 := \text{symdec } enc3, PK$ ;
]_S

```

```

GPSK : Peer  $\equiv$  [
  receive  $SNon\hat{c}.\hat{S}.CSL$ ;
  new  $PNon\hat{c}$ ;
   $MK := \text{prg } PSK$ ;
   $InputString := PNon\hat{c}.\hat{P}.SNon\hat{c}.\hat{S}$ ;
   $SK := \text{kdf1 } InputString, MK$ ;
   $PK := \text{kdf2 } InputString, MK$ ;
   $enc1 := \text{symenc } pl1, PK$ ;
   $mac1 := \text{mac } \hat{P}.\hat{S}.PNon\hat{c}.SNon\hat{c}$ .
     $CSL.CSS.enc1$ ;
  send  $\hat{P}.\hat{S}.PNon\hat{c}.SNon\hat{c}$ .
     $CSL.CSS.enc1.mac1$ ;

  receive  $PNon\hat{c}.SNon\hat{c}.\hat{S}.CSL$ .
     $enc2.mac2$ ;
   $\text{verifymac } mac2, PNon\hat{c}.SNon\hat{c}$ .
     $CSL.enc2, SK$ ;
   $pl2 := \text{symdec } enc2$ ;
   $enc3 := \text{symenc } pl3$ ;
   $mac3 := \text{mac } enc3, SK$ ;
  send  $enc3.mac3$ ;
]_P

```

5.3 EAP-GPSK Security Properties

Setup Assumption. To establish security properties of the *EAP-GPSK* protocol, we assume that the Server \hat{S} and the Peer \hat{P} in consideration are both honest and the only parties which know the corresponding shared PSK . However, we allow all other principals in the network to be potentially malicious and capable of reading, blocking and changing messages being transmitted according to the symbolic model of a network attacker.

$$\phi_{setup} \equiv \text{Honest}(\hat{P}) \wedge \text{Honest}(\hat{S}) \wedge (\text{Has}(X, PSK) \supset \hat{X} = \hat{S} \vee \hat{X} = \hat{P})$$

Security Theorems. The secrecy theorem for *EAP-GPSK* establishes that the signing and encryption keys SK and PK should not be known to any principal other than the peer and the server. For server \hat{S} and peer \hat{P} , this property is formulated as $SEC_{gpsk}(S, P)$ defined as:

$$SEC_{gpsk}(S, P) \equiv (\text{Has}(X, PK) \vee \text{Has}(X, SK)) \supset (\hat{X} = \hat{S} \vee \hat{X} = \hat{P})$$

Theorem 1 (Secrecy). *On execution of the server role, key secrecy holds. Similarly for the peer role. Formally, $EAP\text{-}GPSK \vdash SEC_{pk,sk}^{server}, SEC_{pk,sk}^{peer}$, where*

$$\begin{aligned} SEC_{pk,sk}^{server} &\equiv [\mathbf{GPSK} : \mathbf{Server}]_S SEC_{gpsk}(S, P) \\ SEC_{pk,sk}^{peer} &\equiv [\mathbf{GPSK} : \mathbf{Peer}]_P SEC_{gpsk}(S, P) \end{aligned}$$

Proof Sketch. We skip the rigorous formal proof here, but the proof intuition is as follows: PSK is assumed to be known to \hat{P} and \hat{S} only. The keys SK, PK are derived by using PSK in a key derivation function (MK could be a truncation of PSK or generated by application of a PRG to PSK , according to the length needed). The honest parties use SK, PK as only encryption or signature keys - none of the payloads are derived by a kdf application. This is the intuition why SK, PK remain secrets. A rigorous proof would employ a stronger induction hypothesis and induction over all honest party actions. \square

The authentication theorem for *EAP-GPSK* establishes that on completion of the protocol, the principals agree on each other's identity, protocol completion status, the cryptographic suite list and selection, and each other's nonces. The authentication property for *EAP-GPSK* is formulated in terms of matching conversations [42]. The basic idea of matching conversations is that on execution of a server role, we prove that there exists a role of the intended peer with a corresponding view of the interaction. For server \hat{S} , communicating with client \hat{P} , matching conversations is formulated as $AUTH_{gpsk}(S, P)$ defined below:

$$\begin{aligned} AUTH_{gpsk}(S, P) \equiv & (\text{Send}(S, msg1) < \text{Receive}(P, msg1)) \wedge \\ & (\text{Receive}(P, msg1) < \text{Send}(P, msg2)) \wedge \\ & (\text{Send}(P, msg2) < \text{Receive}(S, msg2)) \wedge \\ & (\text{Receive}(S, msg2) < \text{Send}(S, msg3)) \end{aligned}$$

Theorem 2 (Authentication). *On execution of the server role, authentication holds. Similarly for the peer role. Formally, $EAP\text{-}GPSK \vdash AUTH_{peer}^{server}, AUTH_{server}^{peer}$, where*

$$\begin{aligned} AUTH_{peer}^{server} &\equiv [\mathbf{GPSK} : \mathbf{Server}]_S \exists \eta. P = (\hat{P}, \eta) \wedge AUTH_{gpsk}(S, P) \\ AUTH_{server}^{peer} &\equiv [\mathbf{GPSK} : \mathbf{Peer}]_P \exists \eta. S = (\hat{S}, \eta) \wedge AUTH_{gpsk}(S, P) \end{aligned}$$

Proof Sketch. The formal proof in PCL is in Appendix A. We describe the proof intuition here. We needed to add two new axioms **MAC0** and **VMAC** (also written in Appendix A) to the extant PCL proof system in order to reason about macs. Axiom

MAC0 says that anybody computing a mac on a message m with key k must possess both m and k . Axiom **VMAC** says that if a mac is verified to be correct, it must have been generated by a `mac` action.

$AUTH_{peer}^{server}$: The Server verifies the `mac1` on `msg2` to be a mac with the key SK . By axiom **VMAC**, it must have been generated by a `mac` action and by **MAC0**, it must be by someone who has SK . Hence by secrecy, it is either P or S and hence in either case, an honest party. It is an invariant of the protocol that a mac action on a message of the form $\hat{X}.\hat{Y}.XNonce.YNonce.CSL.CSS.enc$ is performed by a thread of \hat{X} , captured by Γ_1 - hence it must be a thread of \hat{P} , say P . Also using Γ_1 we prove that P received the first message and generated nonce $PNonce$ and sent it out first in the message `msg2`. From the actions of S , we also have that S newly generated $SNonce$ and sent it out first in `msg1`. Using this information and axioms **FS1**, **FS2**, we can order the receives and sends as described in $AUTH_{peer}^{server}$.

$AUTH_{server}^{peer}$: The Peer verifies the `mac2` on `msg3` to be a mac with the key SK . By axiom **VMAC**, it must have been generated by a `mac` action and by **MAC0**, it must be by someone who has SK . Hence by secrecy, it is some thread of either \hat{P} or \hat{S} and hence in either case, an honest party. It is an invariant of the protocol that a mac action on a message of the form $YNonce.XNonce.\hat{Y}.CSL.enc$, is performed by a thread of \hat{Y} , captured by Γ_2 - hence it must be a thread of \hat{S} , say S .

However this mac does not bind the variables CSS and `enc1` sent in `msg2`. So to ensure that S received the exact same message that P sent, we use Γ_2 to further reason that S verified a mac on a message of the form `msg2` and axioms **VMAC**, **MAC0** again to reason that this mac was generated by threads of \hat{S} or \hat{P} . Now, we can use Γ_1 and the form of `msg2` to reason that a thread of \hat{P} did it which also generated $PNonce$ - hence by **AN1**, it must be P itself. Now we use an invariant stating that a thread generating such a mac does it uniquely, captured by Γ_3 , thus binding CSS , `enc1`. Now we use **FS1**, **FS2** as in the previous proof to establish the order described in $AUTH_{server}^{peer}$. \square

Discussion. The formal proof presented above applies to the case where fresh nonces are generated every time. When the peer uses the same nonce repeatedly until it succeeds in completion we have to use a different form of reasoning to ensure the intended message ordering. Specifically, the predicate $\text{FirstSend}(P, PNonce, \text{msg2})$ does not necessarily hold anymore. However, we can still appeal to the fact, that a MAC must have been generated and sent out before it could be received and verified, in order to order messages. Formalizing this requires the new axiom $VMAC'$:

$$\begin{aligned} \mathbf{VMAC}' \quad & \text{Receive}(X, m2) \wedge \text{Contains}(m2, m') \wedge \text{VerifyMac}(X, m', m, k) \wedge \\ & \neg \text{Mac}(X, m, k) \supset \quad \exists Y, m1. \text{Mac}(Y, m, k) \wedge \text{Contains}(m1, m') \wedge \\ & (\text{Send}(Y, m1) < \text{Receive}(X, m2)) \end{aligned}$$

The proof above uses axioms previously proved sound in the symbolic model. While proofs for some properties of *EAP-GPSK* in the computational model could be carried out in computational PCL ([332]), we currently do not have the technical machinery to prove message ordering as a consequence of using fresh nonces in CPCL.

6 Conclusions

In this paper we analyzed the EAP-GPSK authentication protocol. We found three anomalies: a repairable DoS attack, an anomaly in the derivation of the master key MK, and a potential ciphersuite downgrading attack. While the third anomaly seems unavoidable, proper awareness of an attacker's ability to weaken CSuiteList in *Message 1* should prevent problems from arising.

We found that by flooding the network with fake *Message 1*'s, an attacker can force a peer to re-compute the MAC key SK, causing the peer to be unable to correctly process *Message 3* from a legitimate server. This attack is especially worrisome when considering that GPSK is designed to work on devices with limited memory which can easily be exhausted. We propose a fix that allows the peer to maintain state per server instead of state per message.

We identified an anomaly in the derivation of the master key MK. Specifically, MK was derived using a KDF with constant key 0x00. While this does not provide an obvious way for an attacker to reliably learn session keys, it is better to use a more standard implementation.

We used a finite state verification tool named Mur ϕ to search for new problems which may have arisen from the fixes we proposed. We found none. Finally we proved the fixed protocol correct. The analysis was introduced during the standardization process. Throughout our analysis we discussed the weaknesses and possible solutions with the IETF EMU working group. The changes we suggested have subsequently been adopted by the protocol designers. They have been incorporated in the latest internet draft.

Acknowledgements

We would like to thank the protocol designers Charles Clancy and Hannes Tschofenig as well as the EMU working group chair Joe Salowey for fruitful and enjoyable discussions. We also want to thank Sam Hartman for putting us in contact with the EMU working group. We also thank Adam Barth and Taral Joglekar for helpful input in our initial discussions. Finally, we thank the anonymous reviewers for their useful suggestions and comments.

References

1. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J.: Extensible Authentication Protocol. RFC 3748 (2004)
2. Clancy, T., Tschofenig, H.: EAP Generalized Pre-Shared Key (EAP-GPSK) (work in progress) (2007), <http://www.ietf.org/internet-drafts/draft-ietf-emu-eap-gpsk-05.txt>
3. Datta, A., Derek, A., Mitchell, J.C., Shmatikov, V., Turuani, M.: Probabilistic polynomial-time semantics for a protocol security logic. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 16–29. Springer, Heidelberg (2005)
4. Datta, A., Derek, A., Mitchell, J.C., Warinschi, B.: Computationally sound compositional logic for key exchange protocols. In: Proceedings of 19th IEEE Computer Security Foundations Workshop, pp. 321–334. IEEE, Los Alamitos (2006)

5. Dill, D.L.: The Mur ϕ Verification System. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 147–158. Springer, Heidelberg (1996)
6. Mitchell, J.C., Mitchell, M., Stern, U.: Automated Analysis of Cryptographic Protocols Using Mur ϕ . In: IEEE Symposium on Security and Privacy, pp. 141–151 (1997)
7. Mitchell, J.C., Shmatikov, V., Stern, U.: Finite-State Analysis of SSL 3.0. In: SSYM 1998: Proceedings of the 7th conference on USENIX Security Symposium, 1998, Berkeley, CA, USA, pp. 16–16. USENIX Association (1998)
8. He, C., Mitchell, J.C.: Analysis of the 802.11i 4-Way Handshake. In: WiSe 2004: Proceedings of the 3rd ACM Workshop on Wireless Security, pp. 43–50. ACM, New York (2004)
9. Datta, A., Derek, A., Mitchell, J.C., Roy, A.: Protocol Composition Logic (PCL). *Electron. Notes Theor. Comput. Sci.* 172, 311–358 (2007)
10. Meadows, C.: A model of computation for the NRL protocol analyzer. In: Proceedings of 7th IEEE Computer Security Foundations Workshop, pp. 84–89. IEEE, Los Alamitos (1994)
11. Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, A.: *Modelling and Analysis of Security Protocols*. Addison-Wesley Publishing Co., Reading (2000)
12. Fábrega, F.J.T., Herzog, J.C., Guttman, J.D.: Strand spaces: Why is a security protocol correct? In: Proceedings of the 1998 IEEE Symposium on Security and Privacy, Oakland, CA, pp. 160–171. IEEE Computer Society Press, Los Alamitos (1998)
13. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. *Journal of Computer Security* 6, 85–128 (1998)
14. Bella, G., Paulson, L.C.: Kerberos version IV: Inductive analysis of the secrecy goals. In: Quisquater, J.-J., Deswarte, Y., Meadows, C., Gollmann, D. (eds.) ESORICS 1998. LNCS, vol. 1485, pp. 361–375. Springer, Heidelberg (1998)
15. Butler, F., Cervesato, I., Jaggard, A.D., Scedrov, A.: A Formal Analysis of Some Properties of Kerberos 5 Using MSR. In: Fifteenth Computer Security Foundations Workshop—CSFW-15, Cape Breton, NS, Canada, pp. 175–190. IEEE Computer Society Press, Los Alamitos (2002)
16. Butler, F., Cervesato, I., Jaggard, A.D., Scedrov, A.: Verifying confidentiality and authentication in kerberos 5. In: Futatsugi, K., Mizoguchi, F., Yonezaki, N. (eds.) ISSS 2003. LNCS, vol. 3233, pp. 1–24. Springer, Heidelberg (2004)
17. Cervesato, I., Jaggard, A., Scedrov, A., Tsay, J.K., Walstad, C.: Breaking and fixing public-key kerberos (Technical report)
18. Cervesato, I., Meadows, C., Pavlovic, D.: An encapsulated authentication logic for reasoning about key distribution. In: CSFW-18, IEEE Computer Society, Los Alamitos (2005)
19. Datta, A., Derek, A., Mitchell, J.C., Pavlovic, D.: A derivation system and compositional logic for security protocols. *Journal of Computer Security* 13, 423–482 (2005)
20. Hasebe, K., Okada, M.: Non-monotonic properties for proving correctness in a framework of compositional logic. In: Foundations of Computer Security Workshop, pp. 97–113 (2004)
21. Hasebe, K., Okada, M.: Inferences on honesty in compositional logic for security analysis. In: Futatsugi, K., Mizoguchi, F., Yonezaki, N. (eds.) ISSS 2003. LNCS, vol. 3233, pp. 65–86. Springer, Heidelberg (2004)
22. He, C., Sundararajan, M., Datta, A., Derek, A., Mitchell, J.C.: A modular correctness proof of ieee 802.11i and tls. In: ACM Conference on Computer and Communications Security, pp. 2–15 (2005)
23. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology* 15, 103–127 (2002)
24. Backes, M., Pfizmann, B., Waidner, M.: A universally composable cryptographic library. *Cryptology ePrint Archive, Report 2003/015* (2003)
25. Baudet, M., Cortier, V., Kremer, S.: Computationally Sound Implementations of Equational Theories against Passive Adversaries. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 652–663. Springer, Heidelberg (2005)

26. Canetti, R., Herzog, J.: Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 380–403. Springer, Heidelberg (2006)
27. Herzog, J.: Computational Soundness for Standard Assumptions of Formal Cryptography. PhD thesis, MIT (2004)
28. Adão, P., Bana, G., Scedrov, A.: Computational and information-theoretic soundness and completeness of formal encryption. CSFW 18, 170–184 (2005)
29. Micciancio, D., Warinschi, B.: Soundness of formal encryption in the presence of active adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
30. Warinschi, B.: A computational analysis of the Needham-Schroeder(-Lowe) protocol. In: Proceedings of 16th Computer Science Foundation Workshop, pp. 248–262. ACM Press, New York (2003)
31. Roy, A., Datta, A., Derek, A., Mitchell, J.C., Seifert, J.P.: Secrecy analysis in Protocol Composition Logic. In: Proceedings of 11th Annual Asian Computing Science Conference (to appear, 2006)
32. Roy, A., Datta, A., Derek, A., Mitchell, J.C.: Inductive proofs of computational secrecy. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 219–234. Springer, Heidelberg (2007)
33. Roy, A., Datta, A., Mitchell, J.C.: Formal proofs of cryptographic security of Diffie-Hellman-based protocols. In: 3rd Symposium on Trustworthy Global Computing, TGC 2007. LNCS, vol. 4912, pp. 312–329. Springer, Heidelberg (2008)
34. Roy, A., Datta, A., Derek, A., Mitchell, J.C.: Inductive trace properties for computational security. In: WITS (2007)
35. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (2006)
36. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: STOC 1998: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 209–218. ACM, New York (1998)
37. Durgin, N., Mitchell, J.C., Pavlovic, D.: A compositional logic for protocol correctness. In: Proceedings of 14th IEEE Computer Security Foundations Workshop, pp. 241–255. IEEE, Los Alamitos (2001)
38. Durgin, N., Mitchell, J.C., Pavlovic, D.: A compositional logic for proving security properties of protocols. Journal of Computer Security 11, 677–721 (2003)
39. Datta, A., Derek, A., Mitchell, J.C., Pavlovic, D.: A derivation system for security protocols and its logical formalization. In: CSFW-16, pp. 109–125. IEEE, Los Alamitos (2003)
40. Datta, A., Derek, A., Mitchell, J.C., Pavlovic, D.: Secure protocol composition. In: Proceedings of 19th Annual Conference on Mathematical Foundations of Programming Semantics. Electronic Notes in Theoretical Computer Science, vol. 83 (2004)
41. Datta, A., Derek, A., Mitchell, J.C., Pavlovic, D.: Abstraction and refinement in protocol derivation. In: CSFW-17, pp. 30–45. IEEE, Los Alamitos (2004)
42. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)

A Formal Proofs

New Axioms

MAC0 $\text{Mac}(X, m, k) \supset \text{Has}(X, m) \wedge \text{Has}(X, k)$

VMAC $\text{VerifyMac}(X, m', m, k) \supset \exists Y. \text{Mac}(Y, m, k) \wedge m' = \text{MAC}[k](m)$

Invariants

$$\begin{aligned}
\Gamma_1 &\equiv \text{Mac}(Z, \hat{X}.\hat{Y}.X\text{Nonce}.Y\text{Nonce}.CSL.CSS.enc, K) \supset \\
&\quad \hat{Z} = \hat{X} \wedge (\text{Receive}(Z, Y\text{Nonce}.\hat{Y}.CSL) < \\
&\quad \text{Send}(Z, \hat{X}.\hat{Y}.X\text{Nonce}.Y\text{Nonce}.CSL.CSS.enc.mac)) \wedge \\
&\quad mac = \text{MAC}[K](\hat{X}.\hat{Y}.X\text{Nonce}.Y\text{Nonce}.CSL.CSS.enc) \wedge \\
&\quad \text{FirstSend}(Z, X\text{Nonce}, \hat{X}.\hat{Y}.X\text{Nonce}.Y\text{Nonce}.CSL.CSS.enc.mac) \\
\Gamma_2 &\equiv \text{Mac}(Z, Y\text{Nonce}.X\text{Nonce}.\hat{Y}.CSL.enc, SK) \wedge SK = \text{KDF1}[K](Y\text{Nonce}.\hat{Y}.X\text{Nonce}.\hat{X}) \supset \\
&\quad \hat{Z} = \hat{Y} \wedge \exists CSS', enc1. (\text{Send}(Z, Y\text{Nonce}.\hat{Y}.CSL) < \\
&\quad \text{Receive}(Z, \hat{Y}.\hat{X}.Y\text{Nonce}.X\text{Nonce}.CSL.CSS'.enc1.mac1) < \\
&\quad \text{Send}(Z, Y\text{Nonce}.X\text{Nonce}.CSL.enc.mac)) \wedge \\
&\quad mac1 = \text{MAC}[SK](\hat{Y}.\hat{X}.Y\text{Nonce}.X\text{Nonce}.CSL.CSS'.enc1) \wedge \\
&\quad mac = \text{MAC}[SK](Y\text{Nonce}.X\text{Nonce}.CSL.enc) \wedge \\
&\quad \text{VerifyMac}(Z, mac1, \hat{Y}.\hat{X}.Y\text{Nonce}.X\text{Nonce}.CSL.CSS'.enc1, SK) \wedge \\
&\quad \text{FirstSend}(Z, Y\text{Nonce}, Y\text{Nonce}.\hat{Y}.CSL) \\
\Gamma_3 &\equiv \text{Mac}(Z, \hat{X}.\hat{Y}.X\text{Nonce}.Y\text{Nonce}.CSL.CSS.enc, K) \wedge \\
&\quad \text{Mac}(Z, \hat{X}.\hat{Y}.X\text{Nonce}.Y\text{Nonce}.CSL.CSS'.enc', K) \supset CSS = CSS' \wedge enc = enc'
\end{aligned}$$

Formal Proof of AUTH_{peer}^{server}

$$\begin{aligned}
\text{AA1} \quad [\mathbf{GPSK} : \mathbf{Server}]_S \quad &\text{VerifyMac}(S, mac1, \hat{P}.\hat{S}.P\text{Nonce}.S\text{Nonce}. \\
&\quad CSL.CSS.enc1, SK) \tag{1} \\
SEC_{pk,sk}^{server}, \mathbf{VMAC} \quad &[\mathbf{GPSK} : \mathbf{Server}]_S \exists X. (\hat{X} = \hat{P} \vee \hat{X} = \hat{S}) \wedge \\
&\quad \text{Mac}(X, \hat{P}.\hat{S}.P\text{Nonce}.S\text{Nonce}.CSL.CSS.enc1, SK) \tag{2} \\
\Gamma_1 \quad [\mathbf{GPSK} : \mathbf{Server}]_S \quad &\exists \eta. P_0 = (\hat{P}, \eta) \wedge \\
&\quad \text{Mac}(P_0, \hat{P}.\hat{S}.P\text{Nonce}.S\text{Nonce}.CSL.CSS.enc1, SK) \wedge \\
&\quad \text{Receive}(P_0, msg1) < \text{Send}(P_0, msg2) \wedge \\
&\quad \text{FirstSend}(P_0, P\text{Nonce}, msg2) \tag{3} \\
\text{Inst } P_0 \longrightarrow P \quad &[\mathbf{GPSK} : \mathbf{Server}]_S \text{Mac}(P, \hat{P}.\hat{S}.P\text{Nonce}.S\text{Nonce}.CSL.CSS.enc1, SK) \wedge \\
&\quad \text{Receive}(P, msg1) < \text{Send}(P, msg2) \wedge \\
&\quad \text{FirstSend}(P, P\text{Nonce}, msg2) \tag{4} \\
\mathbf{FS1} \quad [\mathbf{GPSK} : \mathbf{Server}]_S \quad &\text{FirstSend}(S, S\text{Nonce}, msg1) \tag{5} \\
\mathbf{FS2}, (-2, -1) \quad &[\mathbf{GPSK} : \mathbf{Server}]_S (\text{Send}(S, msg1) < \text{Receive}(P, msg1)) \wedge \\
&\quad (\text{Receive}(P, msg1) < \text{Send}(P, msg2)) \wedge \\
&\quad (\text{Send}(P, msg2) < \text{Receive}(S, msg2)) \tag{6} \\
\text{AA4} \quad [\mathbf{GPSK} : \mathbf{Server}]_S \quad &(\text{Receive}(S, msg2) < \text{Send}(S, msg3)) \tag{7} \\
(-2, -1) \quad &AUTH_{peer}^{server} \tag{8}
\end{aligned}$$

Formal Proof of $AUTH_{server}^{peer}$

-
- AA1** $[\mathbf{GPSK} : \mathbf{Peer}]_P \text{VerifyMac}(P, mac2, PNonce.SNonce.\hat{S}.CSL.enc2, SK)$ (9)
- $SEC_{pk,sk}^{server}, \mathbf{VMAC}$ $[\mathbf{GPSK} : \mathbf{Peer}]_P \exists X. (\hat{X} = \hat{P} \vee \hat{X} = \hat{S}) \wedge$
 $\text{Mac}(X, PNonce.SNonce.\hat{S}.CSL.enc2, SK)$ (10)
- $\Gamma_2, (-1)$ $[\mathbf{GPSK} : \mathbf{Peer}]_P \exists \eta. S_0 = (\hat{S}, \eta) \wedge$
 $\exists CSS', enc1'. (\text{Send}(S_0, SNonce.\hat{S}.CSL) <$
 $\text{Receive}(S_0, \hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1'.mac1) <$
 $\text{Send}(S_0, PNonce.SNonce.CSL.enc2.mac)) \wedge$
 $mac1 = MAC[SK](\hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1') \wedge$
 $mac = MAC[SK](PNonce.SNonce.CSL.\hat{S}.enc2) \wedge$
 $\text{FirstSend}(S_0, SNonce, SNonce.\hat{S}.CSL)$ (11)
- $\text{Inst } S_0 \longrightarrow S$ $[\mathbf{GPSK} : \mathbf{Peer}]_P \exists CSS', enc1'. (\text{Send}(S, SNonce.\hat{S}.CSL) <$
 $\text{Receive}(S, \hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1'.mac1) <$
 $\text{Send}(S, PNonce.SNonce.\hat{S}.CSL.enc2.mac)) \wedge$
 $mac1 = MAC[SK](\hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1') \wedge$
 $mac = MAC[SK](PNonce.SNonce.\hat{S}.CSL.enc2) \wedge$
 $\text{VerifyMac}(S, mac1, \hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1', SK) \wedge$
 $\text{FirstSend}(S, SNonce, SNonce.\hat{S}.CSL)$ (12)
- $\text{Inst } CSS', enc1',$ $[\mathbf{GPSK} : \mathbf{Peer}]_P \exists X. (\hat{X} = \hat{P} \vee \hat{X} = \hat{S}) \wedge$
- VMAC, MAC0** $\text{Mac}(X, \hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1', SK)$ (13)
- $\Gamma_1, \mathbf{AA1}, (-1)$ $[\mathbf{GPSK} : \mathbf{Peer}]_P \text{New}(X, PNonce) \wedge \text{New}(P, PNonce)$ (14)
- AN1, (-1)** $[\mathbf{GPSK} : \mathbf{Peer}]_P X = P$ (15)
- AA1, (-3, -1)** $[\mathbf{GPSK} : \mathbf{Peer}]_P \text{Mac}(X, \hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS'.enc1', SK) \wedge$
 $\text{Mac}(X, \hat{P}.\hat{S}.PNonce.SNonce.CSL.CSS.enc1, SK)$ (16)
- $\Gamma_3, (-1)$ $[\mathbf{GPSK} : \mathbf{Peer}]_P CSS' = CSS \wedge enc1' = enc1$ (17)
- $(-2, -1)$ $[\mathbf{GPSK} : \mathbf{Peer}]_P (\text{Send}(S, msg1) < \text{Receive}(S, msg2) < \text{Send}(S, msg3))$ (18)
- FS1** $[\mathbf{GPSK} : \mathbf{Peer}]_P \text{FirstSend}(P, PNonce, msg2)$ (19)
- FS2, (12) -2, -1** $[\mathbf{GPSK} : \mathbf{Peer}]_P (\text{Send}(S, msg1) < \text{Receive}(P, msg1)) \wedge$
 $(\text{Send}(P, msg2) < \text{Receive}(S, msg2)) \wedge$
 $(\text{Receive}(S, msg2) < \text{Send}(S, msg3))$ (20)
- AA4** $[\mathbf{GPSK} : \mathbf{Peer}]_P (\text{Receive}(P, msg1) < \text{Send}(P, msg2))$ (21)
- $(-2, -1)$ $AUTH_{server}^{peer}$ (22)
-

Efficient Device Pairing Using “Human-Comparable” Synchronized Audiovisual Patterns

Ramnath Prasad^{1,*} and Nitesh Saxena²

¹ Microsoft

raprasad@windows.microsoft.com

² Polytechnic University

nsaxena@poly.edu

Abstract. “Pairing” is referred to as the operation of achieving authenticated key agreement between two human-operated devices over a short- or medium-range wireless communication channel (such as Bluetooth, WiFi). The devices are ad hoc in nature, i.e., they can neither be assumed to have a prior context (such as pre-shared secrets) with each other nor do they share a common trusted on- or off-line authority. However, the devices can generally be connected using auxiliary physical channel(s) (such as audio, visual) that can be authenticated by the device user(s), and thus form the basis for pairing.

One of the main challenges of device pairing is the lack of good quality output interfaces (e.g., a speaker, display) as well as receivers (e.g., a microphone, camera) on both devices. In this paper, we present a new pairing scheme that is universally applicable to any pair of devices, supporting all possible pairing scenarios. Our scheme does not require devices to have good transmitters or any receivers, and is based upon the device user(s) comparing short and simple synchronized audiovisual patterns, such as in the form of “beeping” and “blinking”.

1 Introduction

Short- or Medium-range wireless communication, based on technologies such as Bluetooth and WiFi, is becoming increasingly popular and promises to remain so in the future. This surge in popularity brings about various security risks. Wireless communication channel is easy to eavesdrop upon and to manipulate, and therefore a fundamental security objective is to secure this communication channel. In this paper, we will use the term “pairing” to refer to the operation of bootstrapping secure communication between two devices connected with a short-range wireless channel. The examples of pairing, from day-to-day life, include pairing of a WiFi laptop and an access point, a Bluetooth keyboard and a desktop. Pairing would be easy to achieve if there existed a global infrastructure

* Work done while being a Master’s student at Polytechnic University.

enabling devices to share an on- or off-line trusted third party, a certification authority, a PKI or any pre-configured secrets. However, such a global infrastructure is close to impossible to come by in practice, thereby making pairing an interesting and a challenging research problem¹

A recent research direction to pairing is to use an auxiliary physically authenticatable channel, called an out-of-band (OOB) channel, which is governed by humans, i.e., by the users operating the devices. Examples of OOB channels include audio, visual and tactile. Unlike the wireless channel, on the OOB channel, an adversary is assumed to be incapable of modifying messages, however, it can eavesdrop on, and possibly also delay, drop and replay them. A pairing scheme should therefore be secure against such an adversary.

The usability of a pairing scheme based on OOB channels is clearly of utmost importance. Since the OOB channels typically have low bandwidth, the shorter the data that a pairing scheme needs to transmit over these channels, the better the scheme becomes in terms of usability.

Various pairing protocols have been proposed so far. These protocols are generally based on the bidirectional automated device-to-device (d2d) OOB channels. Such d2d channels require both devices to have transmitters and the corresponding receivers. In settings, where d2d channel(s) do not exist (i.e., when at least one device does not have a receiver) and even otherwise, same protocols can be based upon device-to-human (d2h) and human-to-device (h2d) channel(s) instead. Depending upon the protocol, only two d2h channels might be sufficient, such as in case when the user has to perform a very simple operation (such as “comparison”) of the data received over these channels. Clearly, the usability of d2h and h2d channel establishment is even more critical than that of a d2d channel.

The earlier pairing protocols required at least 160 to 80 bits of data to be transmitted over the OOB channels. The simplest protocol [1] involves devices exchanging their public keys over the wireless channel, and authenticating them by exchanging (at least 80-bits long) hashes of corresponding public keys over the OOB channels. The more recent, so-called SAS- (Short Authenticated Strings) based protocols, [5], [7], reduce the length of data to be transmitted over the OOB channels to only 15 bits or so²

Based on the above protocols, a number of pairing schemes with various OOB channels have been proposed. These include schemes based on two bidirectional d2d infra-red channels [1]; two bidirectional d2d visual channels consisting of barcodes and photo cameras [6]; a unidirectional d2d visual channel consisting of blinking LED and video camera plus a unidirectional d2h channel consisting of a blinking LED and a unidirectional h2d channel [10]; two audio/visual d2h channels consisting of MadLib sentences and displayed text [4]. In addition, the SAS protocols trivially yield pairing schemes involving two bidirectional d2h and h2d channels – the user reads 15 bits of data displayed on one device and inputs it

¹ The problem has been at the forefront of various recent standardization activities, see [14].

² For the SAS-based authentication and related prior work, refer to [16].

on the other, and vice versa. Most recently, [15] performed user studies of pairing schemes based on user comparing the data transmitted over two independent d2h SAS channels.

The aforementioned schemes have varying degree of usability and are applicable to different device combinations. However, all the above schemes become inapplicable in pairing scenarios where,

1. both devices do not have good quality transmitters (such as displays, speakers, etc.), and
2. both devices do not have relevant receivers (such as cameras, microphones, etc.).

Notice that the pairing scenarios involving most commodity devices, such as access points, headsets, would fall into the above categories.

A very recent proposal, [11], focuses on pairing two devices with the help of “button presses” by the user. The scheme can be used to pair devices with minimal hardware interfaces (such as an LED and a button) using a SAS protocol. However, as we discuss in the next section and as indicated by the results of [11], the scheme is a bit slow.

In short, the previous pairing schemes are either not applicable or are slow in routinely performed pairing scenarios, such as pairing of a WiFi laptop/PDA/cell phone and an access point, a Bluetooth keyboard and a desktop/laptop/PDA.

Our Contributions. In this paper, we propose a new efficient scheme that is universally applicable to pair any two devices.³ Such a universality of a pairing scheme with respect to devices, in our opinion, would improve both security as well as usability over time. Our scheme can use the existing SAS protocols and does not require devices to have good transmitters or any receivers, e.g., only a pair of LEDs are sufficient. The scheme involves users comparing very simple audiovisual patterns, such as “beeping” and “blinking”, transmitted as simultaneous streams, forming two synchronized d2h channels.⁴ We tested our scheme with the three combinations we call *Beep-Beep*, *Blink-Blink* and *Beep-Blink*. Our test results indicate that the *Blink-Blink* and *Beep-Blink* combinations perform very efficiently and robustly, with the former being preferable. However, we discard the *Beep-Beep* combination because it turns out to be quite inefficient and error-prone from our initial testing.

The *Blink-Blink* and *Beep-Blink* combinations are quite efficient in pairing scenarios where both devices do not have good quality transmitters and only at most one device has a relevant receiver. The *Blink-Blink* combination can typically only be used for pairing two similar devices (such as a Bluetooth headset and a cell phone, two laptops, two cell phones) that are physically very close by

³ Our proposal is also equally applicable to establish unidirectional authentication, such as between a printer and a laptop.

⁴ We notice that in an independent result [9], the authors present a scheme similar to our “blinking” scheme, aimed at the detection of “evil twin” access points. The two schemes, however, differ significantly in their implementation and thus in terms of the underlying user experience. We discuss these differences in the next section.

and can be aligned properly with each other. The Beep-Blink combination, on the other hand, can be used for any two devices (generally, one of the devices being paired has an audio transmitter and the other has LEDs). Moreover, the Beep-Blink combination is applicable irrespective of the extreme proximity of devices. In other words, the Beep-Blink combination is also suitable for pairing, e.g., a wall-mounted access point with other devices.

We anticipate that, even in scenarios where devices have good transmitters and also have receivers, the Blink-Blink and Beep-Blink combinations would perform better than most prior solutions [6], [15]. Intuitively, this is due to the reason that the schemes in [15] require the users to perform two operations, reading and comparing, while our schemes only require comparing; whereas the scheme in [6] requires the users to handle specialized equipments, such as cameras, which they might not have used previously. Of course, these are mere expectations. Only a detailed comparative study of all these schemes (which is an item for our future work) can give a clear insight into their applicability among an average user population.

In addition to the single user pairing scenarios, our proposal is also equally and efficiently applicable to scenarios where two users pair their individual devices, such as their cell phones, laptops.

Organization. The rest of the paper is organized as follows. In Section 2, we review the prior pairing schemes. In Section 3, we describe the security model and summarize relevant protocols. In Section 4, we present our scheme, followed by the description of its design, implementation and performance in Section 5.

2 Related Work

There exists a significant amount of prior work on the general topic of pairing.

In their seminal work, Stajano, et al. [13] proposed to establish a shared secret between two devices using a link created through a physical contact (such as an electric cable). In many settings, however, establishing such a physical contact might not be possible, for example, the devices might not have common interfaces to do so or it might be too cumbersome to carry the cables along. Balfanz, et al. [1] extended this approach through the use of infrared as a d2d channel – the devices exchange their public keys over the wireless channel followed by exchanging (at least 80-bits long) hashes of their respective public keys over infrared. The main drawback of this scheme is that it is only applicable to devices equipped with infrared transceivers.

Another approach taken by a few research papers is to perform the key exchange over the wireless channel and authenticate it by requiring the users to manually and visually compare the established secret on both devices. Since manually comparing the established secret or its hash is cumbersome for the users, schemes were designed to make this visualization simpler. These include Snowflake mechanism [3] by Levienet et al., Random Arts visual hash [8] by Perrig et al. etc. These schemes, however, require high-resolution displays and are thus only applicable to a limited number of devices, such as laptops.

Based on the pairing protocol of Balfanz et al. [1], McCune et al. proposed the “Seeing-is-Believing” (SiB) scheme [6]. SiB involves establishing two unidirectional visual d2d channels – one device encodes the data into a two-dimensional barcode and the other device reads it using a photo camera. Since the scheme requires both devices to have cameras, it is only suitable for pairing devices such as camera phones.

Goodrich, et al. [4], proposed a pairing scheme based on “MadLib” sentences. This scheme also uses the protocol of Balfanz et al. The main idea is to establish a d2h channel by encoding the data into a MadLib sentence. Device *A* encodes the hash of its public key into a MadLib sentence and transmits this over a d2h channel (using a speaker or a display); device *B* encodes the hash of the (received) public key from device *A* into a MadLib sentence and transmit this over a d2h channel (using a speaker or a display); the user reads and compares the data transmitted over the two d2h channels, and vice versa. The scheme, as proposed in the paper, requires four d2h channels and the user needs to perform two comparisons. This is quite slow and tedious for the user. One can trivially improve the scheme by using a slightly modified protocol, the one where devices exchange the hash (of size at least 160-bits) of the concatenation of both public keys, after exchanging their public keys. The modified scheme would then require only one user comparison. Note that, however, the scheme is not applicable to pairing scenarios where one of the devices does not have a display or a speaker.

Note that the previously described schemes, with trivial modifications, can (and should) all be based upon one of the SAS protocols [5], [7]. Since the SAS protocols require only 15-bits of data to be transmitted over the OOB channel, such a migration will immensely improve the efficiency as well as the usability of these schemes.

Saxena et al. [10] proposed a new scheme based on visual OOB channel. The scheme uses one of the SAS protocols [5], and is aimed at pairing two devices *A* and *B* (such as a cell phone and an access point), only one of which (say *B*) has a relevant receiver (such as a camera). First, a unidirectional d2d channel is established by device *A* transmitting the SAS data, e.g., by using a blinking LED and device *B* receiving it using a video camera. This is followed by device *B* comparing the received data with its own copy of the SAS data, and transmitting the resulting bit of comparison over a d2h channel (say, displayed on its screen). Finally, the user reads this bit transmitted and accordingly indicates the result to device *A* by transmitting a bit over an h2d input channel.

A very recent proposal, [11], focuses on pairing two devices with the help of “button presses” by the user. The scheme described in the paper is based upon a protocol that first performs an unauthenticated Diffie-Hellman key agreement and then authenticate the established key using a short password. Such a short password can be agreed upon between the two devices via three variants using button presses. The first variant involves the user simultaneously pressing buttons on both devices within certain intervals and each of these intervals are used to derive 3-bits of the password (and thus with 5 button presses, the user is able to inputs the same password on both devices). In the other two variants, one

device picks up a short password, encodes each 3-bit block of the password into the delay between consecutive flashing of the device’s screen or its vibration. As one device flashes or vibrates, the user presses the button on the other device thereby transmitting the password from one device to another. One drawback with the scheme, as described in [11], is that its security is based upon the secrecy of the agreed upon password. At least the button presses and the flashing of the screen can possibly be recorded by a video camera and therefore, the secrecy of the password is not guaranteed. The scheme, however, can easily be based upon a SAS protocol in a straight-forward manner and be used for pairing devices which do not have good transmitters or receivers. Assuming that both devices have an LED and a button each, we can have them transmit their SAS values by blinking of the LED (on one device) and pressing of button (on the other) and vice versa. Unfortunately, this would be quite slow – to transmit a 15-bit SAS value, it will take about a minute in each direction (see the results of the scheme called “D-To-B” in [11]; users can possibly not perform simultaneous “blink-press” faster than 3-4 seconds). One could apply the protocol variant of Saxena et al. [10] to avoid transmission of SAS in the other direction thereby reducing the execution time to close to a minute.

Uzun et al. [15] carry out a comparative usability study of simple pairing schemes. They consider pairing scenarios where devices are capable of displaying 4-digits of SAS data. In what they call the “Compare-and-Confirm” approach, the user simply reads and compares the SAS data displayed on both devices. The “Select-and-Confirm” approach, on the other hand, requires the user to select a 4-digit string (out of a number of strings) on one device that matches with the 4-digit string on the other device. The third approach, called “Copy-and-Confirm”, requires the user to read the data from one device and input it onto the other. These schemes are undoubtedly simple, however, the results of [15] seem to indicate that Select-and-Confirm and Copy-and-Confirm are error prone.

In [12], authors consider the problem of pairing two devices which might not share any common wireless communication channel at the time of pairing, but do share only a common audio channel.

We notice that in an independent result [9], the authors present a scheme similar to the “blinking” scheme that we present in this paper. The scheme of [9] is aimed at the detection of “evil twin” access points in cafs, airport lounges, etc. The two schemes, however, differ significantly in their implementation and therefore in terms of user experience. Firstly, in the scheme of [9], the user controls the time period during which she compares each bit of the SAS data, by pressing and releasing a button on her device. Our scheme, on the other hand, is automatic in that this time period is a pre-determined experimental value. Secondly, in [9], the user’s device needs to trigger the display of next bit on the other device by sending it a signal over the wireless channel. This requires k such signals for a k -bit long SAS and the user needs to verify whether or not these signals are delayed, dropped or injected. This is unlike our scheme, where only one synchronization signal is sent between the two devices.

3 Communication and Security Model, and Applicable Protocols

The pairing protocols are based upon the following communication and adversarial model [16]. The devices being paired are connected via two types of channels: (1) a short-range, high-bandwidth bidirectional wireless channel, and (2) auxiliary low-bandwidth physical OOB channel(s). Based on device types, the OOB channel(s) can be device-to-device (d2d), device-to-human (d2h) and/or human-to-device (h2d). An adversary attacking the pairing protocol is assumed to have full control on the wireless channel, namely, it can eavesdrop, delay, drop, replay and modify messages. On the OOB channel, the adversary can eavesdrop, delay, drop, replay and re-order messages, however, it can not modify them. In other words, the OOB channel is assumed to be an authenticated channel. The security notion for a pairing protocol in this setting is adopted from the model of authenticated key agreement due to Canneti and Krawczyk [2]. In this model, a multi-party setting is considered wherein a number of parties simultaneously run multiple/parallel instances of pairing protocols. In practice, however, it is reasonable to assume only two-parties running only a few serial/parallel instances of the pairing protocol. For example, during authentication for an ATM transaction, there are only two parties, namely the ATM machine and a user, restricted to only three authentication attempts. The security model does not consider denial-of-service (DoS) attacks. Note that on wireless channels, explicit attempts to prevent DoS attacks might not be useful because an adversary can simply launch an attack by jamming the wireless signal.

To date, two three-round pairing protocols based on short authenticated strings (SAS) have been proposed [7], [5]. For the sake of completeness, we depict the protocol of [7] in Figure 1.

In a communication setting involving two users restricted to running three instances of the protocol, these SAS protocols need to transmit only k ($= 15$) bits of data over the OOB channels. As long as the cryptographic primitives used in the protocols are secure, an adversary attacking these protocols can not win with a probability significantly higher than 2^{-k} ($= 2^{-15}$). This gives us security equivalent to the security provided by 5-digit PIN-based ATM authentication.

Recall that the pairing scheme that we propose in this paper requires the users to “compare” the data transmitted over two d2h channels. Our scheme can be based on the existing SAS protocols. This is because in these protocols, the SAS messages are computed as a common function of the public keys and/or random nonces exchanged during the protocol, and therefore the authentication is based upon whether the two SAS messages match or not [7] [5]. The security of these SAS protocols is based upon different cryptographic assumptions. For example, [7] is based upon the random oracle model (ROM), while [5] is not. Moreover, these protocols have different computational requirements. Therefore, based upon the security requirements and underlying devices, our scheme can resort to either of the SAS protocols as desired.

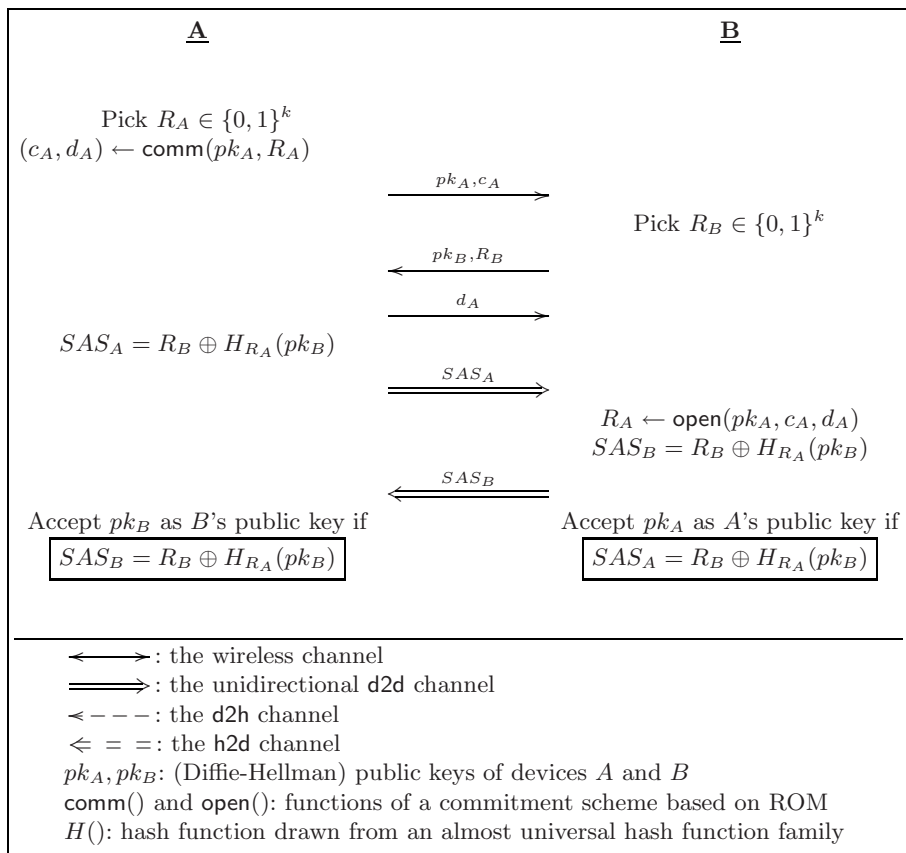


Fig. 1. The SAS protocol of [7]

4 “Human-Comparable” Audiovisual Patterns

Our primary goal is to support pairing scenarios in which both devices do not have good transmitters (e.g., displays) and only at most one device has a receiver (e.g., cameras). Recall that these scenarios include pairing of a laptop and an access point, a keyboard and a desktop, a cell phone and an access point, etc. Our idea is to make use of the existing SAS protocols and implement two synchronized d2h channels to transmit the SAS strings in the form of streams that the user can compare. The synchronization can be achieved by one device signalling the other over the wireless channel. Since this synchronization signal can possibly be tampered with by the adversary, each d2h channel also consists of an “END” marker indicating the end of the respective SAS string and the two END markers also need to be compared by the user. Such d2h channels can be implemented using the following audiovisual combinations.

1. **“Beep-Beep”**. This combination requires both devices to have audio transmitters (such as basic speakers or “beepers”). The two devices encode their respective SAS strings using a sound denoted by “S1” – a ‘1’ bit corresponds to the sound S1 and a ‘0’ bit to a “silence” for a certain period. The two devices also encode the END markers using a *distinct* sound denoted by “S2”. The user listens to both devices and determines if the two play two sounds S1 and S2 in synchronization with each other or not. In other words, if S1 on one device is not played with S1 on other device or if S2 on one device is not played with S2 on the other, the user indicates a “failure”.

2. **“Blink-Blink”**. This combination requires both devices to have visual transmitters, the simplest of which are LEDs. In cases where devices have good displays, one could use the whole or a part of the screen as a transmitter. The two devices encode their respective SAS strings into blinking of a green LED – a ‘1’ bit corresponds to a “blink” period and a ‘0’ bit to an “off” period. The two devices also encode the END markers using the glowing of a red LED. The user looks at the two devices and determines if the green LEDs on two devices blink in synchronization with each other and if the red LEDs glow together. In other words, if the green LED on one device does not blink with the green LED on other device or if the red LED on one device does not glow with the red LED on the other, the user indicates a “failure”.

3. **“Beep-Blink”**. This combination requires one device to have an audio transmitter and the other to have a visual transmitter. One device A encode its SAS string using sound S1 and the END marker using sound S2, and the other device B encodes its SAS string into blinking of a green LED and the END marker by glowing a red LED. The user listens to device A while looking at device B and determines if S1 is played in synchronization with the blinking of the green LED and if S2 is played with the glowing of the red LED. In other words, if S1 on device A is not played with the green blinking LED on device B or if S2 on device A is not played with the glowing of the red LED on device B, the user indicates a “failure”.

The security of our schemes is equivalent to the security of the underlying SAS protocol under the assumption that the user does not commit any errors. This brings us to the design, implementation and usability evaluation of our schemes.

5 Experimentation and Testing

We describe the design and implementation of our pairing schemes based on the Beep-Beep, Blink-Blink and Beep-Blink combinations, and their experimental usability study.

5.1 Design and Implementation

Our goal was to design the Beep-Beep, Blink-Blink and Beep-Blink combinations in a manner that can be used on most devices and that the users find simple

and easy to perform. To this end, we chose to realize the “beeping” with simple sounds, a “system beep” and a “buzz”, which can be easily distinguished even amidst some noise, and the “blinking” with LEDs.

A pairing scheme, in its entirety, consists of three phases: (1) the device discovery phase, wherein the devices exchange their identifiers over the wireless channel, prior to communicating, (2) the pairing protocol execution phase, wherein the devices execute the desired pairing protocol over the wireless channel, and (3) the authentication phase, where the devices, using the OOB channels, authenticate the messages exchanged during the previous phase. For the sake of our experimentation, we skipped the first two phases and concentrated on the third phase, because our main goal was to test the feasibility of the way we intended to implement the OOB channels, i.e., using the Beep-Beep, Blink-Blink and Beep-Blink combinations. As mentioned previously, our pairing scheme can be built on top of the SAS protocols [7], [5].

Let us assume that we want to pair two devices A and B . Given that A and B already performed the device discovery and protocol execution phases over the wireless channel, our job is now reduced to A and B encoding the 15-bits of their respective SAS data SAS_A and SAS_B into beeping or blinking, and transmitting it in a synchronized fashion for the user to compare. This encoding should enable the user to easily identify both the good cases, i.e., when $SAS_A = SAS_B$, and also the bad ones, i.e., when $SAS_A \neq SAS_B$.

To achieve synchronization, we simply have one device sending a synchronization signal S to the other device over the wireless channel. The devices encode and start transmitting their respective SAS data right after sending and receiving the bit S . Note that this synchronization signal can possibly be modified, delayed or dropped (either maliciously or otherwise), possibly fooling the users into accepting non-matching SAS strings (for example, strings $SAS_A = “010010”$ and $SAS_B = “100100”$, will appear to be equal to the user if the synchronization signal is delayed by a bit). To counter this, the end of each SAS string is indicated by an END marker, which can be easily distinguished from the beeping and blinking of the SAS strings and compared by the users enabling them to detect any synchronization errors.

Encoding for “beeping”. A ‘1’ bit in the SAS string is signalled using a “system beep”, whereas a ‘0’ bit is signalled using a “silence” for a certain period of time. The END marker is signalled using a distinctively sounding “buzz”. Every bit signal is followed by a brief “sleep interval”. Note that the “human-comparison” is integral to our scheme and it is important that users are able to identify two distinct bit signals. The sleep interval is inserted for this purpose. The time required to compare two SAS strings is inversely proportional to the duration of the sleep interval – the shorter the sleep interval, the faster the comparison, and vice versa. Based on the Beep-Beep, Blink-Blink and Beep-Blink combinations, an optimal range for the sleep interval needs to be determined through experiments. This range needs to be optimal with respect to the comfort level of a typical user and with respect to the time taken for comparison of encoded data. Figure 2 illustrates the encoding process using a sleep interval of 500 msec.

Sleep Interval	← 500ms →	← 500ms →	← 500ms →	← 500ms →	← 500ms →
Bits	1	1	0	0	END
Bit Signal	beep	beep	silence	silence	buzz
<hr/>					
Sleep interval	← 500ms →	← 500ms →	← 500ms →	← 500ms →	← 500ms →
Bits	1	1	0	0	END
Bit Signal	green blink	green blink	off	off	red blink

Fig. 2. Encoding for “beeping” and “blinking” using a sleep interval of 500 msec

Encoding for “blinking”. We use two LEDs, one green and one red, connected to the data pin of the parallel port of a desktop. On receiving a ‘1’ bit in the SAS string, at the data pin, a voltage is applied at that particular data pin and the green LED “glows” on receiving this voltage. The voltage stays on unless it is explicitly cleared. Through our experiments, we found out that the “blinking” is more suitable than the “glowing”. So to modify glowing to blinking, in our implementation, when the bit a ‘1’, a voltage is applied to the pin for a fixed time interval *a*, followed by another fixed time interval *b* where voltage to the data pin is cut off. As in the encoding using system beep, we call the time interval *a + b* as the sleep interval. For example, on a ‘1’ bit, the LED glows for 80 msec and it stays off for the next 420 msec; on a ‘0’ bit, the LED stays off for the entire duration of 500 msec. See Figure 2 for the encoding process using a sleep interval of 500 msec. Again, the optimal values for *a*, *b* and thus for the sleep interval will be determined through experiments. The END marker is similarly implemented using a red LED.

Implementation. For our experiments, we used a Dell machine 1.2Ghz running windows and a Laptop Compaq AMD 64 bit Turion 1.7GHZ processor machine also running windows. In our experiments, the Dell machine simulates a device which only has a transmitter in the form of two LEDs, one green and one red. To enable blinking feature on Dell machine, we connect external LEDs to the data pins of the parallel port. We use C programming for our implementation and make use of the Visual Studio environment. We use the winsock libraries to establish sockets for communication over a wireless 802.11b channel between the Dell machine and the Compaq machine configured in the ad hoc mode. In our implementation, we configure a fixed port to listen for incoming device pairing requests. The Compaq machine is a device that initiates the pairing with the Dell machine. The Compaq machine has inbuilt speakers, that are used to play out the sounds “beep” and “buzz”.

5.2 Usability Testing

Once the implementation for our schemes was complete, we started doing the most critical part, i.e., the user testing. We tested our scheme with **21 subjects**. Being at a University campus, our subjects were young, enthusiastic group open

to new ideas. They belonged to a wide array of background. All of them were given a brief overview of our pairing schemes and they were also made aware of the possible scenarios where one could make use of pairing. Each user was briefed about the experimental setup comprising of the two devices, and was explained what they are supposed to do while testing the Beep-Beep, Blink-Blink and Beep-Blink combinations.

Our primary goal for the user testing was to figure out if the users are easily able to identify the good cases, i.e., when the two signals match, and more importantly, the bad ones, i.e., when the two signals mismatch. In other words, we wanted to know how often do the users commit, if at all, the *safe errors* (i.e. false positives, or identifying a match as a mismatch), and the *fatal errors* (i.e. false negatives, or identifying a mismatch as a match), following the terminology of [15].

The Set-up and Test Cases. The tests for the Beep-Beep, Blink-Blink and Beep-Blink combinations were carried out in an office room of our University, where volunteering subjects helped us perform 15 test cases that we prepared for each combination. 11 of these test cases were designed to test for the match or mismatch in SAS strings and 4 were designed to test for the synchronization delays.

Our tests comprised of 15 different pairs of 21-bit long strings running as different instances of our written programs. 11 of these pairs were intended to test for match or mismatch of the SAS data (with some padding in the beginning) assuming no synchronization delays (i.e., the END markers were always synchronized). The remaining 4 pairs of strings were intended to test for the synchronization delays (i.e., the END markers were not synchronized). In order to determine the fatal errors, we grouped the strings based upon the following types of mismatch. A “single bit” mismatch denotes a single bit mismatch in the SAS string; a “multiple bit” mismatch denotes a multiple bit mismatch in the SAS string; a “single bit-END marker” mismatch denotes a single bit mismatch in the SAS string in conjunction with a mismatch in the END marker (an example being strings “111111” and “111110”, which appear to be matching in presence of a single bit delay if the user misses the mismatch in the first bit and if no END marker is in place. Identifying a mismatch in the first few bits is a common mistake committed by users as we will see next from our initial tests.); and a “multiple bit-END marker” mismatch denotes a multiple bit mismatch in the SAS string in conjunction with a mismatch in the END marker (an example being strings “010010” and “100100”, which appear to be matching in presence of a single bit delay). The order in which these test cases were administered to the users was randomized to prevent users from learning as they proceeded.

The testing for the Beep-Beep combination was done on two Compaq laptop machines. For the Beep-Blink combination the testing was done with one Compaq laptop machine and one Dell desktop machine, simulating, e.g, an access point, with two LEDs connected to a data pin of its parallel port. To test the Blink-Blink case, on the other hand, we simply connected four LEDs (two green and two red) to four data pins of the parallel port of the Dell desktop machine. This

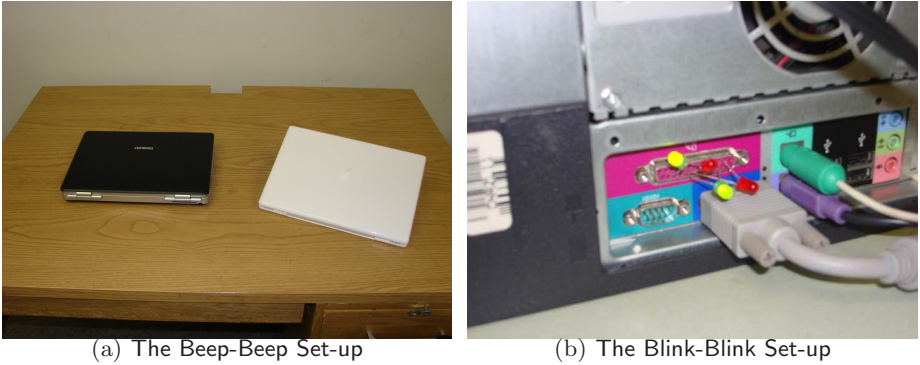


Fig. 3. First two scenarios for the experimental set-up

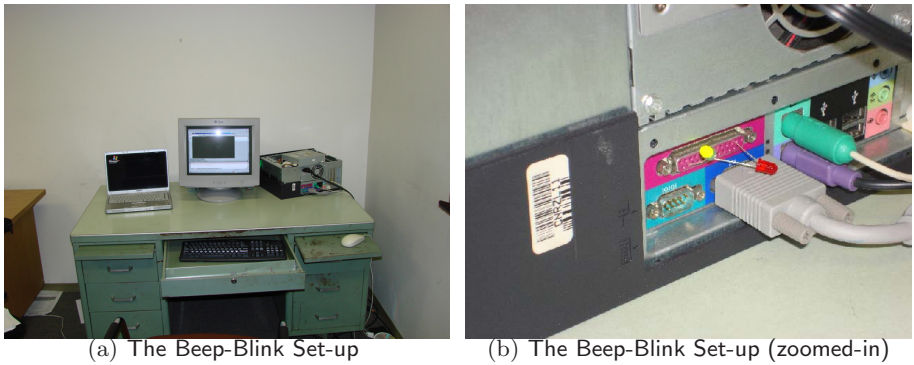


Fig. 4. The third scenario for the experimental set-up

simulated two close by devices with LEDs. See Figures 4 and 3 for some pictures from our experimental test set-up.

Users had to start the process by clicking a button on the Compaq laptop machine, which transmits to the Dell machine the synchronization bit S . Right after, both machines start signalling their respective SAS string by blinking/beeping according to the bit pattern, based on the Beep-Beep, Blink-Blink and Beep-Blink combination being tested.

Some Initial Tests. From some initial tests that we ran, we found out the following. The subjects commit fatal errors when there is a single bit mismatch between the two SAS strings and that the error rate increases when we reduce the sleep interval to anywhere between 200 – 300 msec. Moreover, the subjects commit fatal errors when the 1st bit in the two SAS strings differ.

Immediately analyzing these errors, we concluded that our subjects did not have any learning phase during the signaling. Another important factor was that a subject had to click start one device and almost immediately shift his/her focus

onto both devices simultaneously. If the user is delayed in doing so, then he/she misses the first bit and is thus prone to committing the fatal error. To counter this occurrence of fatal error, we prepend a learning phase of five bits added as an “inner pad” to each SAS string. All the padding bits were 1’s at both devices. For obvious reasons, the use of all 1’s as padding bits was preferred to all 0’s or a combination of 0’s and 1’s. This gave the subjects a learning time before each test. It turns out that including this learning phase of using the inner pad improved the accuracy of our schemes to a great extent. However, it should be noted that subjects were not told about the padding until the last test was carried out by the subject. By including a inner pad to SAS string, we also reduce the bit mismatch to occur somewhere mid string or at the end. Our results show that users were able to clearly identify bit mismatch for these strings with a very high degree of accuracy.

Optimal Values for the sleep interval. Recall that we need to determine the values for the sleep interval, which are optimal with respect to the overall execution time as well to the user comfort level. Through our experiments, we determined the following values for the first 20-bits of the strings being compared, that we will use in our test cases. For the Beep-Beep combination, 300 – 500 msec range was optimal. The Blink-Blink combination fared well with 300 – 800 msec sleep interval, with 300 msec corresponding to 80 msec of a “glow” and 220 msec of an “off”; 500 msec corresponding to 150 msec of a “glow” and 350 msec of an “off”; 800 msec corresponding to 300 msec of a “glow” and 500 msec of an “off”. The sleep interval for the Beep-Blink combination was in the range 300 – 500 msec, with blinking of 300 msec corresponding to 80 msec of a “glow” and 220 msec of an “off”; 400 msec corresponding to 80 msec of a “glow” and 320 msec of an “off”; 500 msec corresponding to 80 msec of a “glow” and 420 msec of an “off”. In each case, the sleep interval for the “blinking” END marker was set to 800 msec, corresponding to 300 msec of a “glow” and 500 msec of an “off”. This higher value was chosen for the users to be able to easily identify any mismatch in the END markers.

The Test Results and their Interpretation. The results⁵ of our user testing for the Beep-Beep, Blink-Blink and Beep-Blink combinations are depicted in Tables 1, 2 and 3, respectively.

Table 1. Responses of 21 users when tested for the Beep-Beep combination

Safe Errors			Fatal Errors			
Sleep Interval	Error Rate	Execution Time	Type of Mismatch	Sleep Interval	Error Rate	Execution Time
500 ms	6/21	10.8sec	single bit	500 ms	16/21	10.8sec
500 ms	5/21	10.8sec	multiple bit	500 ms	11/21	10.8sec

Table 2. Responses of 21 users when tested for the Blink-Blink combination

Safe Errors			Fatal Errors			
Sleep Interval	Error Rate	Execution Time	Type of Mismatch	Sleep Interval	Error Rate	Execution Time
300 ms	0/21	6.8 sec	single bit	300 ms	3/21	6.8 sec
500 ms	0/21	10.8 sec	single bit	500 ms	1/21	10.8 sec
800 ms	0/21	16.8 sec	single bit	800 ms	0/21	16.8 sec
800 ms	0/21	16.8 sec	single bit-END marker	800 ms	0/21	16.8 sec
300 ms	0/21	6.8 sec	multiple bit	300 ms	0/21	6.8 sec
500 ms	0/21	10.8 sec	multiple bit	500 ms	0/21	10.8 sec
800 ms	0/21	16.8 sec	multiple bit	800 ms	0/21	16.8 sec
800 ms	0/21	16.8 sec	multiple bit-END marker	800 ms	0/21	16.8 sec

Table 3. Responses of 21 users when tested for the Beep-Blink combination (given one matching learning instance)

Safe Errors			Fatal Errors			
Sleep Interval	Error Rate	Execution Time	Type of Mismatch	Sleep Interval	Error Rate	Execution Time
300 ms	0/21	6.8 sec	single bit	300 ms	2/21	6.8 sec
400 ms	0/21	8.8 sec	single bit	400 ms	1/21	8.8 sec
500 ms	0/21	10.8 sec	single bit	500 ms	0/21	10.8 sec
500 ms	0/21	10.8 sec	single bit-END marker	500 ms	0/21	10.8 sec
300 ms	0/21	6.8 sec	multiple bit	300 ms	0/21	6.8 sec
400 ms	0/21	8.8 sec	multiple bit	400 ms	0/21	8.8 sec
500 ms	0/21	10.8 sec	multiple bit	500 ms	0/21	10.8 sec
500 ms	0/21	10.8 sec	multiple bit-END marker	500 ms	0/21	10.8 sec

We tested the Beep-Beep combination for bit mismatch in SAS strings only (and not in END marker mismatch). The results indicate quite high (around 50-75%) fatal error rates, as well as high safe error rate (around 30%). This is due to the fact that in order to identify matching as well as mismatching strings, i.e., to determine whether the devices beep together or not, the user needs to be aware of the pitch at which the two devices beep and also of the exact orientation of the devices. Gauging the pitches of the devices is not easy for the users and it requires them to concentrate heavily. Moreover, increasing the sleep interval duration did not improve the error rates (this is why we are only showing the results for 500 msec sleep interval). Clearly, as the results indicate, a mismatch in a single bit is even harder for the users to identify than mismatch in multiple bits. Based on these poor results, we decided not to pursue any further tests (i.e., for the END marker mismatch) for the Beep-Beep combination.

⁵ No user reaction timings are taken into account.

For the Blink-Blink combination, the results are quite promising. There are no safe errors, irrespective of the sleep interval duration. The users do commit some fatal errors (around 3-15%) in cases of single bit mismatch in the SAS strings when the sleep intervals are 500 msec and 300 msec. However, with a sleep interval of 800 msec, there are absolutely no errors at all – the users are easily able to detect correctly a match or a mismatch in the SAS strings as well as a match or a mismatch in the END markers. The whole process took 16.8sec to complete. Since all four LEDs were attached to the same parallel port in our testing, the users were able to focus upon two of them simultaneously and as the sleep interval duration was increased, they felt more comfortable and were able to accurately identify the mismatch.

From some of our initial tests for the Beep-Blink combination, we found out that the users commit a great number of fatal errors. Except a few users, all of them could not detect correctly the very first mismatching instance consisting of a single bit mismatch in the SAS string in conjunction with a single bit delay. However, we also observed that when the users were given the very first instance as a matching instance, they could very accurately identify any other mismatching instances. This implies that a matching instance at the beginning acts as a learning instance for the users that trains them to correctly detect any errors later on. Our tests show that given one matching instance of learning, the only errors were the single bit fatal errors (around 3-9%) with the sleep interval of 400 msec and 300 msec. The sleep interval of 500ms yielded absolutely no errors and an execution time of only 10.8sec. Our results are intuitive – it is somewhat hard for the users to compare two blinking LEDs with two sounds right away, however, given one learning instance, the users get attuned to the process and correctly perform the comparison.

User Feedback. After the users finished their tests, we asked them about their order of preference among the three combinations. Out of the 21 users, 19 users preferred the Blink-Blink combination over the Beep-Blink combination. Clearly, the Beep-Beep combination was everyone’s last choice.

6 Discussion and Conclusion

With the results in hand, we discuss the applicability of our proposal. We decide to discard the Beep-Beep combination, and choose the Beep-Blink combination with a sleep interval of 500 msec and an execution time of 10.8sec (given one learning instance) and the Blink-Blink combination with a sleep interval of 800 msec and an execution time of 16.8sec.

The Blink-Blink and Beep-Blink combinations are quite efficient solutions for the pairing scenarios where both devices do not have good quality transmitters and only at most one device has a relevant receiver. The Blink-Blink combination can typically be used for pairing two similar devices (such as a Bluetooth headset and a cell phone, two laptops, two cell phones) that are physically very close by and can be aligned properly with each other. The Beep-Blink combination, on the other hand, can be used for any two devices (generally, one of the devices being

paired has an audio transmitter and the other has LEDs). Moreover, the Beep-Blink combination is applicable irrespective of the extreme proximity of devices. In other words, the Beep-Blink combination is also suitable for pairing, e.g., a wall-mounted access point with other devices. Of course, there is a downside to the Beep-Blink combination in that it requires the users be trained with one matching instance beforehand. However, such a training can easily be administered to the user on one of user's own devices, for example, in the form of a simulation of the Beep-Blink combination on user's cell phone. Notice that the user is anyway generally explained various steps involved in setting up its devices, using brochures or CDs.

Overall, we rate Blink-Blink over Beep-Blink because it does not require any learning and was preferred by most users in our tests. Furthermore, in noisy environments, Blink-Blink is anyway a better choice than Beep-Blink.

We hope that, even in scenarios where devices have good transmitters and also have receivers, the Blink-Blink and Beep-Blink combinations would perform better than most prior solutions [6], [15]. This is due to the reason that the schemes in [15] require the users to perform two operations, reading and comparing, while our schemes only require comparing; whereas the scheme in [6] requires the users to handle specialized equipments, such as cameras, which they might not have used previously. We expect that our schemes and the scheme of [4] (modified using one of the SAS protocols and with one device displaying a MadLib sentence while the other "speaking" it out) might be comparable. Of course, these are mere expectations. Only a detailed comparative study of all these schemes (which is an item for our future work) can give a clear insight into their applicability among an average user population. In our future work, we would also like to compare the Blink-Blink and Beep-Blink combinations with the schemes of [11] and [9]. Recall that, similar to Blink-Blink and Beep-Blink, these two schemes also require only minimal interfaces on the devices.

In conclusion, we believe that our Blink-Blink and Beep-Blink schemes can be adopted in practice. This is so not only because of their efficiency and robustness as indicated by our test results and their universal applicability, but also because of another subtle, yet obvious, reason. Notice that "beeping" and "blinking", naturally and universally so, often serve the purpose of alerting humans in day to day life, such as when used in car indicators, in fire alarms, on ambulances, fire brigades and police vehicles, and so on. The use of beeping and blinking, as proposed, is therefore a natural approach to realize a critical security operation, such as pairing.

Acknowledgements

We would like to thank Gene Tsudik for discussions on the subject of the paper and N. Asokan for his comments on the previous versions. We would also like to thank the anonymous reviewers for their helpful comments.

References

1. Balfanz, D., Smetters, D., Stewart, P., Wong, H.C.: Talking to strangers: Authentication in ad-hoc wireless networks. In: Network and Distributed System Security Symposium (NDSS) (2002)
2. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2001)
3. Goldberg, I.: Visual Key Fingerprint Code (1996), <http://www.cs.berkeley.edu/iang/visprint.c>
4. Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and clear: Human-verifiable authentication based on audio. In: International Conference on Distributed Computing Systems (ICDCS) (2006)
5. Laur, S., Asokan, N., Nyberg, K.: Efficient mutual data authentication based on short authenticated strings. In: IACR Cryptology ePrint Archive: Report 2005/424 (2005)
6. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-is-believing: Using camera phones for human-verifiable authentication. In: IEEE Symposium on Security and Privacy (S&P) (2005)
7. Pasini, S., Vaudenay, S.: Sas-based authenticated key agreement. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 395–409. Springer, Heidelberg (2006)
8. Perrig, A., Song, D.: Hash visualization: a new technique to improve real-world security. In: International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC 1999) (1999)
9. Roth, V., Polak, W., Rieffel, E., Turner, T.: Simple and effective defenses against evil twin access points. In: ACM Conference on Wireless Network Security (WiSec), short paper (2008)
10. Saxena, N., Ekberg, J.-E., Kostianen, K., Asokan, N.: Secure device pairing based on a visual channel. In: IEEE Symposium on Security and Privacy (S&P), short paper (2006)
11. Soriente, C., Tsudik, G., Uzun, E.: Beda: Button-enabled device association. In: International Workshop on Security for Spontaneous Interactio (IWSSI) (2007)
12. Soriente, C., Tsudik, G., Uzun, E.: Hapadep: Human asisted pure audio device pairing. In: IACR Cryptology ePrint Archive: Report 2007/093 (2007)
13. Stajano, F., Anderson, R.J.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Malcolm, J.A., Christianson, B., Crispo, B., Roe, M. (eds.) Security Protocols 1999. LNCS, vol. 1796, Springer, Heidelberg (2000)
14. Suomalainen, J., Valkonen, J., Asokan, N.: Security associations in personal networks: A comparative analysis. In: European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS) (2007)
15. Uzun, E., Karvonen, K., Asokan, N.: Usability analysis of secure pairing methods. In: Usable Security (USEC). LNCS, vol. 4886, pp. 307–324. Springer, Heidelberg (2007)
16. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)

PUF-HB: A Tamper-Resilient HB Based Authentication Protocol

Ghaith Hammouri and Berk Sunar

Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609-2280
{hammouri, sunar}@wpi.com

Abstract. We propose a light-weight protocol for authentication of low-power devices. Our construction PUF-HB merges the positive qualities of two families of authentication functions. PUF represents physically unclonable functions and fulfills the purpose of providing low-cost tamper-resilient challenge-response authentication. On the other hand, the Hopper Blum (HB) function provides provable cryptographic strength against passive adversaries. By building on an earlier proof of the security of HB⁺ by Katz et al. [1], we rigorously prove the security of the proposed scheme against active adversaries. While the active adversary model does not include man-in-the-middle attacks, we show that a previously successful man-in-the-middle attack proposed for HB⁺, does not carry to PUF-HB.

Keywords: HB⁺, PUF, tamper resilience, provable security.

1 Introduction

Low cost and power efficient devices are essential for the next generation ubiquitous networks. By utilizing devices ranging from wireless sensor nodes to RF identification devices (RFIDs) and smartcards, these networks are envisioned to support a large number of applications carrying out diverse tasks. It is clear that low-cost and lightweight security schemes are absolutely essential for the adoption of ubiquitous networks. To further highlight the security requirements of these networks, we observe that despite their diversity, the deployed devices have three things in common: they store sensitive information, they transmit data through unprotected wireless networks, and typically an adversary will have physical access to the device due to the nature of the applications. It is essential that sensitive information is safely stored and communicated. However, since most of the pervasive devices are passive devices with limited power sources and with stringent constraints imposed on their computational resources, implementing protection measures on these low-cost devices becomes a challenge. Furthermore, attacks that circumvent the data channel and instead exploit the so called side-channels [43] are posing a major threat to constrained devices. These attacks are classified into two groups. Passive attacks solely observe side-channels (e.g. computation time, power consumption, electromagnetic emanation, temperature attacks etc.) to deduce internal secrets from leaked side-channel profiles.

In contrast, in active attacks the attacker may also inject faults during the computation [5]. Not surprisingly, active attacks are more powerful and are more difficult to prevent. Furthermore, active attacks have also been demonstrated to be useful in circumventing countermeasure against side-channel leakage. Thus, building tamper-proof hardware is crucial for securing devices against passive and active side-channel attacks.

The recently proposed Physically Unclonable Functions (PUFs) provide a promising alternative approach for achieving tamper-resilience [6]. A PUF is a physical pseudo-random function which may be implemented by exploiting the small variances of the wire and gate delays that are unique for every single integrated circuit (IC), even if they are logically identical. These variances depend on highly unpredictable factors, which are mainly caused by the inter-chip manufacturing variations. Hence, given the same input, the PUF circuit will return a different output on different ICs. Additionally, if the PUFs are manufactured with high precision, any major external physical influence will change the PUF function and render the device non-functional. These features are indeed very attractive for any low-cost authentication scheme hoping to achieve tamper-resilience. However, all PUF based authentication schemes proposed so far rely on collecting challenge-response pairs which are unique for every PUF. These challenge-response pairs have to be stored in a database [6], so that the data can be retrieved and used to authenticate any given PUF. It is not hard to see that this solution becomes less practical when the number of devices deployed increases. Another major problem faced by the delay-based PUF design is that unless augmented by traditional cryptographic schemes (e.g. hash functions) or non-linearization techniques [7,8,12] it will not be secure against simple modeling attacks.

In complement to PUFs the HB-based authentication schemes provide a security reduction. In [13] Hopper and Blum (HB) proposed the first HB authentication scheme. The HB protocol is indeed promising for simplifying the authentication process and significantly reducing the power consumption of pervasive networks. An additional major advantage of the HB protocol is that its security is based on the hardness of the learning parity with noise (LPN) problem. The LPN problem is known to be NP-hard [25]. Unfortunately, as pointed out in [13] the HB scheme is weak against active adversaries. In [14], Juels and Weis proposed a hardened version of the HB protocol labeled HB⁺ which resists active attacks in the detection based model.¹ HB⁺ was shown to be secure [14,11] in the detection based model. In [18], the authors demonstrated a man-in-the-middle attack for breaking HB⁺.

In this paper we merge the PUF authentication scheme along with the HB based authentication protocol to produce a hybrid protocol which enjoys the advantages of both schemes while improving the level of security. We propose an authentication scheme which is tamper resilient, and at the same time provably secure against active attacks in the detection based model. In addition, the

¹ In this model, a flag is raised whenever a tag fails to authenticate for a large number of times.

presented protocol resists the man-in-the-middle attacks proposed so far for the HB^+ scheme. From the PUF perspective, our protocol is the first PUF based authentication scheme which is provably secure. For our security proof we closely follow the proof presented in [1].

The remainder of the paper is organized as follows. In the next section we introduce PUFs and present a mathematical model. In Section 3 we give a review of the previously proposed HB based authentication protocols. In Section 4 we define our notation and describe our protocol. The security reduction is presented in Section 5. In Section 6 we describe the man-in-the-middle attack and show that the proposed protocol resists it. In Section 7 we discuss hardware implementation and tamper resilience properties of the proposed scheme. Finally, we present the conclusions in Section 8.

2 Physically Unclonable Functions

A PUF is a challenge-response circuit which takes advantage of interchip variations to achieve tamper-resilience. The idea behind a PUF is to have the same logical circuit produce a different output depending on the actual implementation parameters of the circuit. The variations from one circuit implementation to another are not controllable and are directly related to the physical aspects of the environment. These aspects include temperature, pressure levels, electromagnetic waves and quantum fluctuations. Therefore, two identical logical circuits sitting right next to each other on the same die might still have quite different input-output behavior due to the nature of a PUF.

The reason one might seek to explore this property is to prevent an attacker from cloning the function. Additionally, because of the high level of sensitivity of these physical functions it becomes virtually impossible for an attacker to accurately reproduce the hardware. Another major advantage of the sensitivity of the PUF is to prevent physical attacks on the system. Trying to tap into the circuit will cause a capacitance change and therefore change the output of the PUF. Removing the outer layer of the chip will have a permanent effect on these circuit parameters and again, will change the output of the PUF circuit. Roughly summarized, we can say that a well-built PUF device will be physically tamper-resilient up to its sensitivity level. We would like to note here that one of the major advantages of PUF circuits is their lightweight nature. Using a recent proposal which utilizes tri-state buffers to construct PUF circuits [11], one can show that for an n bit PUF about $2n$ gates is sufficient.

A delay-based PUF [6] is a $\{0, 1\}^k \rightarrow \{0, 1\}$ mapping, that takes a k -bit challenge a and produces a single output bit r . As shown in Figure 1 the delay-based PUF circuit consists of k stages of switches. Each switch has two input and two output bits in addition to a control bit. If the control bit of the switch is 0, the two inputs are directly passed to the outputs through a straight path. If on the other hand, the control bit to the switch is 1, the two input bits are switched before being passed as output bits. Therefore, the control bit of each switch will decide the path taken by the input signals. The k switches are serially connected

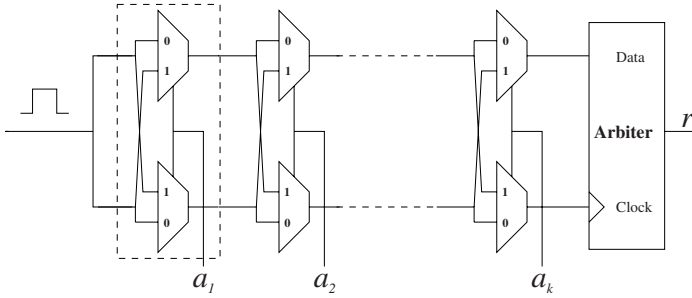


Fig. 1. A basic delay-based PUF circuit

so that the output of each switch is connected to the input of the following switch. The two outputs of the last switch are connected to a flip-flop, which is called the *arbiter*. The two inputs to the first switch are connected to each other, and then connected to a pulse generator.

We briefly describe the operation of PUFs as follows. When a challenge a is received each of its k bits is used as the control bit to one of the PUF switches. Next, a pulse is generated and sent to the first switch. The pulse will break into two pulses entering the upper and lower inputs of the first switch. Depending on the control bit of the first switch the two signals will either travel in a straight path or they will switch locations. Although the paths taken by the signals have been designed to have the same length, due to physical variations the two paths will have a small mismatch. This mismatch will have a different value for each of the two possible paths of the two signals. Therefore, the two pulses will acquire a time delay between them which is dependent on the control bit of the first switch. The same argument applies for the rest of the k switches. Each challenge a will impose a different path on the k switches. Consequently, the total delay mismatch between the two signals will be a function of a . The job of the arbiter at the end of the switch chain is to indicate which signal arrives first. Recall that a flip-flop has two inputs, the data input and the clock input. When the clock input observes a rising edge the data input is captured at the output of the flip-flop. In a PUF setting, if the signal connected to the data input of the arbiter arrives first, the output of the arbiter will be 1. Otherwise, the output will be 0.

In [6] the authors derive a linear delay model for the delay-based PUF. The model is represented by an equation which gives the total delay difference between the two pulses in terms of the challenge bits and the parameters of the PUF circuit. Let D_H represent the total delay of the pulse that enters the data input of the arbiter and D_L be the delay of the pulse that enters the clock input. The mathematical model derived in [6] shows that the difference between the two propagation paths D_H and D_L can be expressed algebraically as

$$\delta = D_L - D_H = \sum_{i=1}^k (-1)^{p_i} y_i + y_{k+1}$$

where $p_i = a_i \oplus a_{i+1} \oplus \dots \oplus a_k$ such that a_i is the i^{th} bit of a . This relation can also be described using $(P = Ua)$ where $P = [p_1, \dots, p_k]$ and a are represented as column vectors. U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2. Here the y_i parameter denotes the imbalance in the signal propagation paths in i^{th} stage of the PUF circuit.² The condition for the output bit of the arbiter circuit, r , can be written as $\delta + T_s > 0 \rightarrow r = 1$ and $\delta + T_s < 0 \rightarrow r = 0$, where T_s denotes the setup time for the arbiter flip-flop. Now, let $Y = [y_1, \dots, y_{k+1}]$ we can compute the output of the PUF using the following function,

$$r = \text{PUF}_Y(a) = \text{sign} \left(\sum_{i=1}^k (-1)^{p_i} y_i + y_{k+1} \right) \quad (1)$$

Where $\text{sign}(x) = 1$ if $x > 0$, and 0 if $x < 0$. When the argument in $\text{sign}(x)$ is zero the output becomes a random bit. In fact, the random output bit will in practice be observed for a slightly larger window of mismatch values. In these cases the PUF is said to be *metastable*. We will shortly address this issue by introducing the noise parameter ϵ . It is important to note that the delay variations y_i will depend on the fabrication process of the PUF circuit. Therefore, one would expect these parameters to follow a normal distribution [10]. In particular, the y_i values will follow a Gaussian distribution of mean zero, and a fixed variance. Without loss of generality, we can normalize these values and assume they belong to a normal distribution of mean 0 and variance 1.

The fact that the PUF function can be represented using a linear inequality means that given a sufficient number of challenge-response pairs $(a_{(i)}, r_{(i)})$ for a single PUF,³ an attacker will be able to model the system using linear programming [27,28,12] or machine learning [29,6] algorithms. This observation seems to completely undermine the idea behind a device labeled unclonable. Although the delay parameters y_i are not measurable, a simple PUF circuit will leak information about these values through its challenge-response pairs. For theoretical and experimental results on PUF modeling, the reader is referred to [8,12,11]. As proposed in [12], in this paper we take advantage of the modelability of the simple PUF circuit. We provide an overall scheme which prevents an attacker from calculating the delay parameters, while at the same time allowing the owner of the circuit to take advantage of this property. We further elaborate on this point in Section 7. For the remainder of the paper we will utilize the PUF model and assume that the y_i parameters fully characterize the operation of the PUF circuit.

It should be noted that even if the best techniques are used to model a PUF circuit, there will always be a level of error in the developed model. This is

² The delay model derived here is slightly different from that given in [6], particularly the y_i parameters differ with a factor of two.

³ For brevity the setup time T_s can be merged with the last delay parameter y_{k+1} .

⁴ In this paper we use subscripts with parenthesis to denote different strings, e.g. $a_{(i)}$, and we use subscripts without parenthesis to denote bits of the same string, e.g. a_i .

due to multiple reasons. First, the thermal noise will cause slight fluctuations in the internal delay parameters y_i . Second, although the system is modeled with a linear equation, there will always be some source of non-linearity which will affect the system. Moreover, the two signals propagating inside a PUF circuit will sometimes enter a race condition such that the decision made by the arbiter will be random [9]. This will typically happen when the total delay difference between the two signals is less than the resolution of the arbiter. When the PUF enters such a state the circuit is said to be metastable. As a result, any PUF device can only be modeled up to a certain level of accuracy, i.e. the model will always mispredict some response bits. The best modeling schemes tested so far were shown to reduce the misprediction rate to as low as 3% [8]. Note that the 3% misprediction rate is obtained by implementing error reduction techniques such as majority voting. For the construction in this paper we utilize the built-in noise to secure the HB component. In our notation we denote the amount of error that exists in a PUF circuit model by ϵ .

Before we conclude this section, we introduce the following notation. For any bit v ,

$$\text{PUF}_{Y,v}(a) = \left\{ \begin{array}{ll} \text{sign} \left(\sum_{i=1}^k (-1)^{p_i} y_i + y_{k+1} \right), & v = 0 \\ \text{sign} \left(\sum_{i=1}^k (-1)^{\overline{p_i}} y_i + y_{k+1} \right), & v = 1 \end{array} \right\}, \quad (2)$$

where $\overline{p_i}$ is the complement of p_i [5]. This notation will simplify the derivations in the following sections.

3 The HB Family

The HB authentication schemes base their security on the hardness of the LPN problem. In this section we give a quick review of the LPN problem and the different HB authentication schemes, focusing on HB and HB⁺. For a certain $\epsilon \in (0, \frac{1}{2})$ the LPN problem is denoted by LPN $_{\epsilon}$, and defined as follows.

Definition 1 ([1]). *Given n random binary k -bit strings $a_{(j)}$ and the bits $z_{(j)} = a_{(j)} \cdot x \oplus \nu$ for some $x \in \{0, 1\}^k$, where $a \cdot b$ denotes the binary inner product between a and b , $\nu = 1$ with probability ϵ and 0 with probability $1 - \epsilon$, then find the binary string x .*

The LPN problem is known to be NP-hard [25]. In [13], the authors show that the LPN problem is log-uniform and even hard to approximate within a ratio of 2. Kearns proved in [26] that the LPN problem is hard in the statistical query model. The best known algorithm to solve the LPN problem is the BKW algorithm [20]. However, there has been a number of improvements on the algorithm with the best running time of $2^{O(k/\log k)}$ [21,22,23].

⁵ The complement may be realized by simply XOR-ing the most-significant bit of a with ν .

In the HB protocol [13], the tag and the reader share a k -bit secret string x . To authenticate the tag, the reader starts sending randomly generated k -bit challenge strings $a_{(j)}$. The tag responds with the bit $z_{(j)} = a_{(j)} \cdot x \oplus \nu$ where the variables are as defined in the LPN problem. The tag and the reader repeat the same step for multiple challenges. Finally, the reader checks to see if the number of errors in the tag's response matches the noise level, and decides to accept or reject accordingly. Note that if the tag's response did not contain noise, then a passive attacker would easily be able to deduce x after collecting k challenge-response pairs using Gaussian elimination. In [13], the authors prove that given an algorithm that predicts $z_{(j)}$ for a random $a_{(j)}$ with some advantage, then this algorithm can be used to solve the LPN_ϵ problem. However, HB is only secure against passive attacks. An active attacker can easily repeat the same challenge multiple times, effectively eliminating the noise and reducing the problem to Gaussian elimination.

To secure the HB protocol against an active attacker the HB^+ protocol was proposed in [14]. In HB^+ the tag and the reader share two k -bit strings x and y . The tag starts the authentication session by sending a random k -bit string $b_{(j)}$. The reader then responds with $a_{(j)}$ just like the HB protocol. Finally the tag responds with $z_{(j)} = a_{(j)} \cdot x \oplus b_{(j)} \cdot y \oplus \nu$, where ν is defined as above. The protocol is proven to be secure against an active attack on the tag (excluding man-in-the-middle attacks). In such an adversary model an attacker is not allowed to obtain final decisions from the reader on whether this authentication session was successful or not. In [14] and [1] the authors show that in this adversary model breaking the HB^+ protocol is equivalent to solving the LPN problem. However, as we pointed out earlier, a simple man-in-the-middle attack was demonstrated on the HB^+ protocol in [18]. Note that in a detection based model this attack will not be successful.

In addition to HB and HB^+ , there has been a number of other variations such as HB^{++} [16], HB-MP [15] and HB^* [24]. All these proposals attempt to prevent man-in-the-middle attacks. In a more recently proposed scheme, i.e. $\text{HB}^\#$ [19], the authors propose a modified version of HB^+ which uses Toeplitz matrices rather than vectors for a shared secret. Under a strong conjecture the scheme is proven secure against a class of man-in-the-middle attacks. In this adversary model which is referred to as *GRS-MIM-model*, the attacker can only modify data transmission from the reader to the tag but not the other way around. This model will protect against the previously mentioned attack.

4 The PUF-HB Authentication Protocol

We start by defining basic notation. In the remainder of the paper we reserve k to denote the security variable. $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ denotes the tag used in the PUF-HB protocol, where $x \in \{0,1\}^k$, $s \in \{0,1\}^{2 \times n}$. We treat s_j as a 2-bit vector. and $Y = [y_1, y_2, \dots, y_{k+1}]$ such that $y_i \in N(0,1)$ and $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2 . The noise parameter is $\epsilon \in (0, \frac{1}{2})$ and n denotes the number of rounds required for authentication. We denote the

reader by $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$, where all the variables are as defined for the tag except l and u which are integers in the range $[0, n]$ such that $l \leq \epsilon \cdot n \leq u$.

With this notation we describe the basic authentication step. In the j^{th} round of the protocol, $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ outputs a randomly generated vector $b_{(j)} \in \{0, 1\}^k$ and sends it to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$. The reader responds with the vector $a_{(j)} \in \{0, 1\}^k$ and $e_{(j)} \in \{0, 1\}^2$. Finally the tag computes $z_{(j)} = b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)}) \oplus \nu$, where \cdot is the binary inner product, s_j are the j^{th} two bit vector of s and $\nu = 1$ with probability ϵ and $\nu = 0$ with probability $1 - \epsilon$. For authentication, this step is repeated n times. In each round the reader checks to see if the tag's response is equal to $b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)})$. If not, the reader marks the response as wrong. At the end of the n^{th} round, the reader authenticates the tag if and only if the number of wrong responses is in the range $[l, u]$.

In general, any entity can interact with the reader and try to impersonate an honest tag. To capture such interaction, let \mathcal{E} be any entity trying to authenticate itself to the reader $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$. Then $PUF-HB(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}) = 1$ iff \mathcal{E} is authenticated by the reader, and is equal to 0 otherwise. The following protocol formalizes this interaction:

Protocol 1: PUF-HB($\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$)

1. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ sets $c = 0$ and $j = 1$.
 2. \mathcal{E} sends $b_{(j)} \in \{0, 1\}^k$ to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$.
 3. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ chooses $a_{(j)} \in \{0, 1\}^k$ and $e_{(j)} \in \{0, 1\}^2$ uniformly at random and sends it to \mathcal{E} .
 4. \mathcal{E} sends $z_{(j)}$ to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$.
 5. If $z_{(j)} \neq b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)})$ then $c = c + 1$.
 6. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ increments j and repeats steps 2 through 5 until $j = n$.
 7. If $l \leq c \leq u$ then $PUF-HB(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}) = 1$, otherwise it equals 0.
-

Provided that \mathcal{E} has no information about x, s or Y , the best probability of being authenticated by the reader will be $\epsilon_s = 2^{-n} \sum_{i=l}^u \binom{n}{i}$. This probability represents the soundness error in the algorithm. As for an honest tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ one can see that with a very high probability $PUF-HB(\mathcal{T}_{(n,x,Y,s,\epsilon)}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}) = 1$. However, the tag's choice to set $\nu = 1$ with probability ϵ is independent in each round of the authentication. Therefore, it will be possible for the tag to introduce a number of errors which is outside the range $[l, u]$. This will result in a failed authentication session. We denote the probability of this incident by ϵ_c , i.e. the completeness error. If we set $l = 0$ then using the Chernoff bound we can produce the following bound, $\epsilon_c < e^{-(\epsilon n)(u/\epsilon n - 1)^2/4}$.

Protocol (1) would work identically if we run it in a parallel fashion. In this case, the n different $b_{(j)}$ vectors sent in Step 2 would be sent in a single step and similarly all the $a_{(j)}$ vectors and $e_{(j)}$ bits sent in Step 3 would also be sent in a single step. Finally all the $z_{(j)}$ bits returned in Step 4 would be sent in a single step. In general, the PUF-HB protocol is almost identical to the HB⁺ protocol. The main difference introduced in the PUF-HB protocol is to substitute the inner product $a \cdot y$ where $y \in \{0, 1\}^k$ with $PUF_{Y,s}(a)$. As we will see in the proof of Theorem \square this substitution will not affect the security features introduced by the

HB⁺ protocol. However, as indicated in the previous sections this substitution will make the tag tamper-resilient, and will simultaneously help resist the known man-in-the-middle attack on the HB⁺ protocol [18].

5 Security Against Active Attacks

In this section, we reduce the security of the PUF-HB protocol in the active attacker model (which does not include man-in-the-middle attacks) to solving the LPN problem. Note that as we pointed out earlier, the proof here closely follows the proof of Katz et al. on the security of the HB⁺ protocol [1]. However, due to the nature of the PUF circuit, a very simple part of the original proof in [1], becomes much more complex in our protocol. Such a difference is a reflection of the change from a simple binary inner product to a PUF operation. For a more elaborate explanation of the proof see [2] where the authors prove security for the parallel execution case with $\epsilon < 1/4$. Also see [17] for a similar proof when $\epsilon < 1/2$. Moreover, in the original paper where the HB⁺ protocol was proposed [14] the authors provide an elegant proof of security against active attacks. The proof in [14] can easily be modified to prove the security of the PUF-HB protocol. However, for simplicity and completeness we use the proof in [2].

We start by quoting the following definitions directly from [2]. Let $A_{x,\epsilon}$ denote an LPN _{ϵ} oracle which outputs a $k + 1$ bit string such that x is chosen uniformly at random from $\{0, 1\}^k$ and $\epsilon \in (0, \frac{1}{2})$. The output of the oracle is the string

$$(a, a \cdot x \oplus \nu) .$$

Where a is chosen uniformly at random from $\{0, 1\}^k$, and $\nu = 1$ with probability ϵ and $\nu = 0$ with probability $1 - \epsilon$. We also define U_{k+1} to be the uniform oracle with outputs from the uniform distribution over $\{0, 1\}^{k+1}$. We say that an algorithm M can (t, q, δ) solve the LPN _{ϵ} problem if

$$\Pr [M^{A_{x,\epsilon}}(1^k) = x] \geq \delta ,$$

provided that M runs in time t and uses q queries to the oracle $A_{x,\epsilon}$. The main theorem of this paper relies on the following lemma originally due to Regev [30] and reproven in [2].

Lemma 1. ([2]) *If there exists an algorithm D making q oracle queries, and running in time t , such that*

$$|\Pr [D^{A_{x,\epsilon}}(1^k) = 1] - \Pr [D^{U_{k+1}}(1^k) = 1]| \geq \delta ,$$

then there exist an algorithm M making $q' = O(q \cdot \delta^{-2} \log k)$ oracle queries and running in time $t' = O(t \cdot k \delta^{-2} \log k)$, such that

$$\Pr [M^{A_{x,\epsilon}}(1^k) = x] \geq \delta/4 .$$

Before we prove the main theorem of the paper we try to give some intuition on why the proof in [11] can be applied to prove the security of the PUF-HB protocol. In addition, we state two technical lemmas which are needed for the proof of the main theorem.

First, note that the function computed in the HB⁺ protocol is $z = b \cdot x \oplus a \cdot y$ whereas the function computed in the PUF-HB protocol is $z = b \cdot x \oplus PUF_{Y,s}(a)$. Moreover, Theorem 1 states that if there exists an algorithm \mathcal{A} which starts by impersonating a reader to interact with an honest tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ (learning phase), and then impersonates a tag to interact with an honest reader (impersonation phase) therefore achieving $\text{PUF-HB}(\mathcal{A}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}) = 1$ with high probability, then algorithm \mathcal{A} can also be used to distinguish between an LPN oracle and a uniform oracle. As implied by Lemma 1, then algorithm \mathcal{A} can be used to solve the LPN problem.

In the learning phase, the algorithm \mathcal{A} will expect to interact with an honest tag. In the proof, we impersonate such a tag, and use the oracle outputs to substitute the $b \cdot x$ part of a tag's response. Note that this part of the response is in common between the HB⁺ protocol and the PUF-HB protocol. Now since we will use an oracle for part of the response, this means that we will have no control over x which will be determined by the oracle. At the same time we will have full control over Y, s (y in the HB⁺ protocol).

In the impersonation phase \mathcal{A} will take the role of a tag interacting with an honest reader, and will therefore attempt to correctly compute the responses $z_{(i)}$. However, the responses of \mathcal{A} will depend on the interaction that took place in the learning phase. If we were successful in providing outputs which were consistent with some x , then \mathcal{A} will be able to provide correct responses in the impersonation phase. This will be the case if the used oracle was an LPN oracle. On the other hand, if the oracle was the uniform oracle, then we will have failed in providing consistent outputs to \mathcal{A} . Consequently, the responses $z_{(i)}$ produced by \mathcal{A} in the impersonating phase will be random. While this presents a technique to distinguish between an LPN oracle and a uniform oracle, nevertheless, we do not have a way to know if the responses $z_{(i)}$ were correct or not. Primarily because we have no knowledge of x . To resolve this issue we run the algorithm and acquire the responses for a set of challenges $\{a_{(i)}^1\}$ along with the $\{e_{(i)}^1\}$ bits, then we rewind the algorithm and acquire a second set of responses for a different set of challenges $\{a_{(i)}^2\}$ and the $\{e_{(i)}^2\}$ bits. Adding the two responses cancels out the effect of x and retains the effect of $\text{PUF}_{Y,e_{(i)}^1 \cdot s_i}(a_{(i)}^1) \oplus \text{PUF}_{Y,e_{(i)}^2 \cdot s_i}(a_{(i)}^2)$ (this would be $a_{(i)}^1 \cdot y \oplus a_{(i)}^2 \cdot y$ for the HB⁺ protocol). With this trick we will have complete knowledge over the remaining variables. Therefore, we can check whether the responses returned by \mathcal{A} were correct or not.

What remains to be shown is that given the inputs $a_{(i)}^1$ and $a_{(i)}^2$ and no other information, the algorithm \mathcal{A} will not be able to predict the output bits $\text{PUF}_{Y,e_{(i)}^1 \cdot s_i}(a_{(i)}^1)$ and $\text{PUF}_{Y,e_{(i)}^2 \cdot s_i}(a_{(i)}^2)$. This is akin to asking the question: How much can be inferred about the output, by only knowing the input. In the HB⁺ protocol this becomes a question of linear independence. However, in the case of PUF-HB the answer becomes much more complicated. This question will be

addressed in Lemma 2 and 3. In general, one can see that the function $PUF_{Y,s}(a)$ (compared to $a \cdot y$) only becomes relevant toward the end of the proof of Theorem 1. In fact, it should be clear that there is a large family of functions that could be used in place of $PUF_{Y,s}(a)$ or $a \cdot y$ while maintaining the security of the protocol. However, our choice of $PUF_{Y,s}(a)$ was mainly motivated by hardware simplicity and tamper resilience. The following Lemma characterizes the correlation between two PUF outputs obtained from different input challenges.

Lemma 2. *Given two k -bit strings $a_{(1)}$ and $a_{(2)}$ chosen independently and uniformly at random from $\{0, 1\}^k$, and $Y_0 = [y_1, \dots, y_{k+1}]$ where $y_i \in N(0, 1)$ for $i = 1, \dots, k$ and $y_{k+1} = 0$, then for $z_{(1)} = PUF_{Y_0}(a_{(1)})$ and $z_{(2)} = PUF_{Y_0}(a_{(2)})$ the probability that $z_{(1)}$ and $z_{(2)}$ are equal is*

$$Pr[z_{(1)} = z_{(2)}] = F_k(d) = 1 - \frac{2}{\pi} \arctan \left(\sqrt{\frac{d}{k-d}} \right). \tag{3}$$

Where d is the Hamming distance between $P_{(1)}$ and $P_{(2)}$ and $(P_{(i)} = Ua_{(i)})$. The strings $a_{(i)}$ are represented as column vectors, U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2.

The proof of Lemma 2 can be found in Appendix A. The next lemma, connects Lemma 2 to the main theorem.

Lemma 3. *Let \mathcal{A} be an adversary who is given n strings $\{a_{(i)}\}_{i=1}^n$, where $a_{(i)}$ is chosen independently and uniformly at random from $\{0, 1\}^k$. \mathcal{A} also knows that s is chosen uniformly at random from $\{0, 1\}^n$ and that $Y_0 = [y_1, \dots, y_{k+1}]$ where $y_i \in N(0, 1)$ for $i = 1, \dots, k$ and $y_{k+1} = 0$. Let $z_{(i)} = PUF_{Y_0,s_i}(a_{(i)})$ then the bits $z_{(i)}$ will be uniform and independent (from the point of view of \mathcal{A}).*

The proof of Lemma 3 is also moved to Appendix A. We are now ready to prove the main theorem.

Theorem 1 (Compare to Theorem 3 in [1]). *If there exists an adversary \mathcal{A} interacting with a tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ in at most q executions of the PUF-HB protocol (possibly concurrently), running in time t such that*

$$Pr[\mathcal{A}^{\mathcal{T}_{(n,x,Y,s,\epsilon)}}(1^k) : PUF-HB(\mathcal{A}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}) = 1] \geq \delta \quad ,$$

then there exist an algorithm D making $q \cdot n$ oracle queries, running in time $O(t)$, and such that

$$|Pr[D^{A_{x,\epsilon}}(1^k) = 1] - Pr[D^{U_{k+1}}(1^k) = 1]| \geq \delta^2 - 2^{-n/2} \sum_{i=0}^{2u} \binom{n/2}{i} - e^{-\frac{n}{8}}$$

Therefore, for any $\epsilon < \frac{1}{8}$ there is an appropriate choice of n, u such that the last two terms become negligible, and thus we can conclude that the PUF-HB protocol is secure assuming the hardness of the LPN_ϵ problem.⁶

⁶ Note that by parameterizing the length ℓ of the strings $s_i \in \{0, 1\}^\ell$ and $e_{(i)} \in \{0, 1\}^\ell$ we may achieve some flexibility in the parameters, i.e. $\epsilon < 0.25 - 2^{-(\ell+1)}$.

Proof. To prove the theorem we show a construction of the algorithm D . As stated by the theorem, D is given access to an oracle returning $(k+1)$ -bit strings which can be broken to (\bar{b}, \bar{z}) , where $\bar{b} \in \{0, 1\}^k$ and $\bar{z} \in \{0, 1\}$. D proceeds as follows:

1. D starts by choosing vectors Y_0 and s such that y_{k+1} is set to 0, and then the k remaining y_i values are chosen from $N(0, 1)$. The bitstring s is chosen uniformly at random from $\{0, 1\}^{2 \times n}$. D runs the algorithm \mathcal{A} which will expect to interact with a PUF-HB tag. In order to impersonate a real tag, D does the following to simulate a basic authentication step: D starts by obtaining a $k + 1$ bit string $(\bar{b}_{(i)}, \bar{z}_{(i)})$ from the oracle, and then sends $\bar{b}_{(i)}$ to \mathcal{A} as the initial b in Protocol 1. \mathcal{A} will reply with a challenge $\bar{a}_{(i)}$ and the bits $\bar{e}_{(i)}$. Next, D computes $z_{(i)} = \bar{z}_{(i)} \oplus \text{PUF}_{Y_0, \bar{e}_{(i)} \cdot s_i}(\bar{a}_{(i)})$ and sends $z_{(i)}$ back to \mathcal{A} . D repeats this step $q \cdot n$ times.
2. In the second phase of the algorithm, \mathcal{A} tries to impersonate an honest tag. Looking at the parallel execution of PUF-HB, \mathcal{A} starts by sending $b_{(1)}, \dots, b_{(n)} \in \{0, 1\}^k$ to a reader. Next, D randomly chooses $a_{(1)}^1, \dots, a_{(n)}^1 \in \{0, 1\}^k$ and $e_{(1)}^1, \dots, e_{(n)}^1 \in \{0, 1\}^2$ and send them back to \mathcal{A} , which will in turn respond with the bits $z_{(1)}^1, \dots, z_{(n)}^1$. D then rewinds \mathcal{A} and sends randomly chosen $a_{(1)}^2, \dots, a_{(n)}^2 \in \{0, 1\}^k$ and $e_{(1)}^2, \dots, e_{(n)}^2 \in \{0, 1\}^2$. \mathcal{A} will respond with $z_{(1)}^2, \dots, z_{(n)}^2$. Note that since the algorithm was rewound the same b values will be sent by \mathcal{A} .
3. D calculates $z_{(i)}^\oplus = z_{(i)}^1 \oplus z_{(i)}^2$ and lets $Z^\oplus = (z_{(1)}^\oplus, \dots, z_{(n)}^\oplus)$. D also computes $\hat{z}_{(i)} = \text{PUF}_{Y_0, e_{(i)}^1 \cdot s_i}(a_{(i)}^1) \oplus \text{PUF}_{Y_0, e_{(i)}^2 \cdot s_i}(a_{(i)}^2)$ and lets $\hat{Z} = (\hat{z}_{(1)}, \dots, \hat{z}_{(n)})$. D outputs 1 iff Z^\oplus and \hat{Z} differ in at most $2u$ positions.

Now we analyze D :

Case 1: If the oracle used by D was the uniform oracle U_{k+1} , then the outputs \bar{z} given to D in step 1 were uniformly distributed and independent of everything. This means that the bits $z_{(i)}$ which D sent back to \mathcal{A} in step 1 were also uniformly distributed and independent of everything. Therefore, by the end of step 1 \mathcal{A} has no information about either Y_0 or s . All \mathcal{A} receives in the second step are the inputs $\{a_{(i)}^1\}_{i=1}^n, \{e_{(i)}^1\}_{i=1}^n$ and $\{a_{(i)}^2\}_{i=1}^n, \{e_{(i)}^2\}_{i=1}^n$. All of these inputs are uniformly and independently distributed. As shown in Lemma 3 each of the two calculated output strings $\{\text{PUF}_{Y_0, e_{(i)}^1 \cdot s_i}(a_{(i)}^1)\}_{i=1}^n$ and $\{\text{PUF}_{Y_0, e_{(i)}^2 \cdot s_i}(a_{(i)}^2)\}_{i=1}^n$ will be uniform over $\{0, 1\}^n$ (from the point of view of \mathcal{A}). However, when we add these two variables and obtain \hat{Z} the individual bits of the output will not always be independent. The affect of the s_i bits will actually cancel out from both terms when $e_{(i)}^1 = e_{(i)}^2$ with probability 0.25. To simplify the proof, we assume in this case that the outputs are completely dependent. Using the Chernoff approximation we bound the probability of observing more than $n/2$ dependent output bits by $e^{-\frac{n}{8}}$. The probability that Z^\oplus and \hat{Z} differ in at most $2u$ positions is exactly $2^{-n/2} \sum_{i=0}^u \binom{n/2}{i}$. We conclude that D outputs 1 in this case with probability at most $2^{-n/2} \sum_{i=0}^{2u} \binom{n/2}{i} + e^{-\frac{n}{8}}$.

Case 2: If D is using the oracle $A_{x,\epsilon}$ for some random x , the simulation provided by D in the first phase will be perfect and therefore \mathcal{A} will be able to impersonate an honest tag with probability at least δ . Let w be the randomness used in the first phase of running \mathcal{A} , then we denote the probability that \mathcal{A} succeeds in impersonating an honest tag for a fixed choice of w by δ_w . Now since we rewind the algorithm, the probability that \mathcal{A} succeeds in both rounds is δ_w^2 . Let $E(\delta_w^2)$ denote the expected value of δ_w^2 over all possible randomness w , then we have

$$E(\delta_w^2) \geq E(\delta_w)^2 = \delta^2,$$

where the square is taken out of the expected value operator using Jensen's inequality. Now assuming that \mathcal{A} succeeds in impersonating an honest tag for both rounds, then we would expect each of the response strings $z_{(1)}^1, \dots, z_{(n)}^1$ and $z_{(1)}^2, \dots, z_{(n)}^2$ to have at most u errors. Therefore Z^\oplus will in turn have at most $2u$ errors.⁷ When we add $z_{(i)}^1$ and $z_{(i)}^2$ to generate z_i^\oplus we are canceling the effect of $b \cdot x$ and therefore we are left with $z_{(i)}^\oplus = \text{PUF}_{Y_0, e_{(i)}^1 \cdot s_i}(a_{(i)}^1) \oplus \text{PUF}_{Y_0, e_{(i)}^2 \cdot s_i}(a_{(i)}^2)$. With the exception of the $2u$ errors in Z^\oplus , the strings \hat{Z} and Z^\oplus are calculating the same output. We conclude that D outputs 1 in this case with probability at least δ^2 . \square

This concludes our security proof, and shows that the PUF-HB protocol is secure against an active attacker provided that the LPN problem is hard to solve.

6 Man-in-the-Middle Attacks

The main weakness of the HB^+ protocol is the man-in-the-middle attack proposed in [18]. Briefly summarized, in this attack an adversary replaces all the challenges $\{a_{(j)}\}_{j=1}^n$ sent from the reader in a single authentication session by $\{a_{(j)} \oplus w\}_{j=1}^n$ where $w \in \{0, 1\}^k$. The attacker knows that the challenges will interact with the secret y through $a_{(j)} \cdot y$. At the end of the n rounds, if the reader authenticates the tag, the adversary can deduce with very high probability that his changes did not affect the responses of the tag, and therefore $w \cdot y = 0$. On the other hand, if the reader rejects the tag, then the adversary will know with a very high probability that $w \cdot y = 1$. Repeating the same attack k times will allow the adversary to collect k linear equations in y . The adversary can use Gaussian elimination to recover the secret y . Similarly, the same attack can be carried out to deduce the other secret string x .

In the PUF-HB scheme this particular man-in-the-middle attack will not succeed due to the non-linearity of the PUF function. From Lemma 2 we know that the only type of correlations that the attacker can exploit are those related to the Hamming distance between the different input strings a . However, we saw in Lemma 3 that with the secret string s the Hamming distance information is

⁷ This worst case happens when the u errors in $z_{(i)}^1$ and $z_{(i)}^2$ affect completely different bits.

masked for a single authentication session. It is still possible that an adversary can exploit Hamming distances between different sessions to launch an attack. Another potential point of weakness is the linearity in the $b \cdot x$ portion of the PUF-HB protocol. To protect against simple attacks exploiting this linearity, a second PUF circuit can be used with the b vector as its input. We label such a protocol PUF²-HB, since it will essentially be identical to Protocol 1, with the only difference in the z bit calculated by the tag, which becomes

$$z_{(i)} = b_{(i)} \cdot x \oplus PUF_{Y_1, s_i}(a_{(i)}) \oplus PUF_{Y_2}(b_{(i)}) \oplus \nu \tag{4}$$

where the shared secret becomes (x, Y_1, Y_2, s) .

7 Hardware Security

In the previous section we discussed the security of the proposed scheme under abstract security models. However, in recent years we have seen numerous side-channel attacks which directly target the hardware implementation. The PUF paradigm was aimed at protecting against active side-channel attacks. In the PUF-HB protocol there are only two strings that are to be stored by the tag: x and s . The secret Y is not really stored since it is part of the characteristics of the circuit itself. In Section 2 we discussed the resilience of PUF circuits against hardware attacks. In particular the PUF prevents an attacker from measuring the y_i parameters directly via a physical measurement. Any major changes to the surrounding temperatures or voltage levels, or any attempt to forcefully read the value of these registers will induce a change to the PUF, therefore changing the identity of the tag.

What is more impressive about the PUF circuit is that it even protects neighboring components. This is achieved by placing all registers containing the secret strings x and s sufficiently close to the PUF circuit. We have experimentally verified these claims on an FPGA hardware implementation of a PUF. Such a level of security ensures that even when the tag itself is compromised, an attacker cannot impersonate this tag by extracting the secrets from the hardware. Naturally, one might be concerned about how that will affect the modelability of the PUF in the pre-deployment phase. Before deployment of the tag, the registers are initialized to their secret values. Afterward, with the knowledge of the secret vectors the reader can develop an accurate model for the PUF circuit. Note that modeling the PUF is even possible in the existence of noise [32]. Hence, the sensitivity of the PUF does not prevent the owner from modeling it.

Finally we would like to underline that PUF-HB is not inherently protected against passive attacks, e.g. Simple Power Analysis and Differential Power Analysis. Although not trivial, side-channel profiles may be utilized to recover the secret values. If passive side-channel attacks are a concern, standard IC level power balancing techniques [33][34][35] must be employed. Although effective, these techniques tend to incur significant area overhead. An alternative approach would be to modify the implementation to balance the power consumption.

8 Conclusion

In this paper we merged the PUF authentication scheme with the HB based authentication protocol, with the goal of producing a hybrid protocol which enjoys the advantages of both schemes while improving the level of security. The main contribution of this paper is the proposed PUF-HB authentication scheme which is tamper resilient, and at the same time provably secure against active attacks in the detection based model. In addition, the proposed protocol resists the known man-in-the-middle attacks against the HB⁺ scheme. From the PUF perspective, the protocol is the first PUF based authentication scheme with a security reduction. From the HB perspective, this is the first tamper-resilient HB-based authentication protocol. From a more practical perspective, the proposed scheme provides low-cost, tamper-resilient, and provably secure authentication.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and CNS-0716306. We would like to thank Dr. Stephen Weis for clarifications regarding the proof of security in [14] and for pointing us to [1], and Prof. William J. Martin for his comments on an earlier draft of the paper.

References

1. Katz, J., Shin, J.S.: Parallel and Concurrent Security of the HB and HB⁺ Protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006)
2. Want, R.: An Introduction to RFID Technology. IEEE Pervasive Computing 5(1), 25 (2006)
3. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
4. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
5. Kulikowski, K.J., Karpovsky, M.G., Taubin, A.: DPA on faulty cryptographic hardware and countermeasures. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 211–222. Springer, Heidelberg (2006)
6. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Delay-based Circuit Authentication and Applications. In: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 294–301 (2003)
7. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: CCS 2002: Proceedings of the 9th ACM conference on Computer and communications security, pp. 148–160. ACM Press, New York (2002)
8. Lee, J.W., Daihyun, L., Gassend, B., S., G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symposium of VLSI Circuits, pp. 176–179 (2004)

9. O'Donnell, C.W., Suh, G.E., Devadas, S.: PUF-based random number generation. Number 481 (November 2004)
10. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. *IEEE Trans. VLSI Syst.* 13(10), 1200–1205 (2005)
11. Ozturk, E., Hammouri, G., Sunar, B.: Physical unclonable function with tristate buffers. In: *The Proceedings of The IEEE International Symposium on Circuits and Systems 2008 – ISCAS* (to appear, 2008)
12. Ozturk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: *PERCOM 2008: Proceedings of the Sixth IEEE International Conference on Pervasive Computing and Communications* (2008)
13. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
14. Juels, A., Weis, S.A.: Authenticating Pervasive Devices with Human Protocols. In: Shoup, V. (ed.) *CRYPTO 2005*. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
15. Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. *Comput. Networks* 51(9), 2262–2267 (2007)
16. Bringer, J., Chabanne, H., Dottax, E.: HB⁺⁺: a Lightweight Authentication Protocol Secure against Some Attacks. In: *SECPERU 2006: Proceedings of the Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, Washington, DC, USA, 2006, pp. 28–33. IEEE Computer Society, Los Alamitos (2006)
17. Katz, J., Smith, A.: Analyzing the HB and HB⁺ protocols in the “large error” case. In: *Cryptology ePrint Archive, Report 2006/326* (2006), <http://eprint.iacr.org/>
18. Gilbert, H., Robshaw, M., Sibert, H.: An Active Attack Against HB⁺ - A Provably Secure Lightweight Authentication Protocol. *IEE Electronic Letters* 41,21, 1169–1170 (2005)
19. Gilbert, H., Robshaw, M., Seurin, Y.: HB[#]: Increasing the Security and Efficiency of HB⁺. In: *Advances in Cryptology: EUROCRYPT 2008*. LNCS, vol. 4965, Springer, Heidelberg (2008)
20. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: *STOC 2000: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pp. 435–440. ACM Press, New York (2000)
21. Fossorier, M., Mihaljevic, M., Imai, H., Cui, Y., Matsuura, K.: A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication. In: *Proc. of INDOCRYPT*, vol. 6, pp. 48–62.
22. Leveil, E., Fouque, P.: An Improved LPN Algorithm. In: De Prisco, R., Yung, M. (eds.) *SCN 2006*. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
23. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subsetsum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) *APPROX 2005 and RANDOM 2005*. LNCS, vol. 3624, pp. 378–389. Springer, Heidelberg (2005)
24. Duc, D., Kim, K.: Securing HB⁺ Against GRS Man-in-the-Middle Attack. In: *Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, January 2007, pp. 23–26 (2007)

25. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.: On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory* 24(3), 384–386 (1978)
26. Kearns, M.: Efficient Noise-Tolerant Learning from Statistical Queries. In: *STOC 1993: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 392–401. ACM Press, New York (1993)
27. Roos, C., Terlaky, T., Vial, J.-P.: *Interior Point Methods for Linear Optimization*, 2nd edn. Springer, Heidelberg (2005)
28. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. *Mathematical Programming* 71(2), 221–245 (1995)
29. Agmon, S.: The relaxation method for linear inequalities. *Canadian J. of Mathematics*, 382–392 (1964)
30. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: *STOC 2005: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp. 84–93. ACM Press, New York (2005)
31. Prudnikov, Y.A., Brychkov, A.P., Marichev: *Integrals and Series*, vol. 2: Special Functions. In: Gordon and Breach (1990)
32. Blum, A., Frieze, A.M., Kannan, R., Vempala, S.: A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica* 22(1/2), 35–52 (1998)
33. Tiri, K., Akmal, M., Verbauwhede, I.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, pp. 403–406 (2002)
34. Toprak, Z., Leblebici, Y.: Low-power current mode logic for improved DPA-resistance in embedded systems. In: *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium*, pp. 1059–1062 (2005)
35. Regazzoni, F., Badel, S., Eisenbarth, T., Grobschadl, J., Poschmann, A., Toprak, Z., Macchetti, M., Pozzi, L., Paar, C., Leblebici, Y., Ienne, P.: A Simulation-Based Methodology for Evaluating the DPA-Resistance of Cryptographic Functional Units with Application to CMOS and MCML Technologies. In: *IC-SAMOS 2007*, pp. 209–214 (2007)

A Appendix 1

Proof of Lemma 2

Proof. First recall from Section 2 that for any string $c = [c_1, \dots, c_n]$ the PUF response is

$$\text{PUF}_Y(c) = \text{sign} \left(\sum_{i=1}^k (-1)^{p_i} y_i + y_{k+1} \right),$$

where $p_i = c_i \oplus \dots \oplus c_k$ and $P = [p_1, \dots, p_k]$, as noted above ($P = Uc$). This mapping is a linear bijection and will therefore preserve uniformity and independence. Since $a_{(1)}$ and $a_{(2)}$ were chosen uniformly at random from $\{0, 1\}^k$, the same can be said about the distribution of the corresponding P vectors $P_{(1)} = [p_1^{(1)}, \dots, p_k^{(1)}]$ and $P_{(2)} = [p_1^{(2)}, \dots, p_k^{(2)}]$. Let $Z(P_{(j)}, Y_0) = \sum_{i=1}^k (-1)^{p_i^{(j)}} y_i$, and let d be the Hamming distance between $P_{(1)}$ and $P_{(2)}$. Without loss of generality we may assume that the different bits of $P_{(1)}$ and $P_{(2)}$ are in the first d

bit positions. Now we can write

$$Z(P_{(1)}, Y_0) = \sum_{i=1}^d (-1)^{p_i^{(1)}} y_i + \sum_{i=d+1}^k (-1)^{p_i^{(1)}} y_i = D_d + S_{k-d}$$

$$Z(P_{(2)}, Y_0) = -D_d + S_{k-d} ,$$

where $D_d = \sum_{i=1}^d (-1)^{p_i^{(1)}} y_i$ and $S_{k-d} = \sum_{i=d+1}^k (-1)^{p_i^{(1)}} y_i$. Note that since $z_{(j)} = \text{sign}(Z(P_{(j)}, Y_0))$, then for $z_{(1)}$ to be equal to $z_{(2)}$ we need $Z(P_{(1)}, Y_0)$ and $Z(P_{(2)}, Y_0)$ to have the same sign. For this, D_d needs to have a smaller magnitude than S_{k-d} which means $|D_d| < |S_{k-d}|$. Therefore

$$\Pr[z_{(1)} = z_{(2)}] = \Pr[|S_{k-d}| - |D_d| > 0] = \Pr[R_d > 0],$$

where $R_d = |S_{k-d}| + (-|D_d|)$. Let $f_R(R_d)$, $f_D(D_d)$ and $f_S(S_{k-d})$ represent the probability distribution function (PDF) for each of the random variables R_d , D_d and S_{k-d} respectively. Each term in the summations making up D_d and S_{k-d} involves one of the bits $p_i^{(1)}$ and the real value y_i . Since $y_i \in N(0, 1)$ has mean zero and is symmetric around the y -axis, we can easily see that multiplying with $(-1)^{p_i^{(1)}}$ will not affect the normal distribution and therefore $(-1)^{p_i^{(1)}} y_i \in N(0, 1)$. Now each of the variables D_d and S_{k-d} is a summation of respectively d and $k-d$ normal distributions $N(0, 1)$. Thus, $f_D(D_d) = N(0, d)$ and $f_S(S_{k-d}) = N(0, k-d)$.⁸ We are interested in the PDF of $-|D_d|$ and $|S_{k-d}|$. These can easily be calculated as follows:

$$f_{|S|}(x) = \begin{cases} 2N(0, k-d) & x > 0 \\ 0 & x \leq 0 \end{cases} = \begin{cases} 2 \frac{e^{-\frac{x^2}{2(k-d)}}}{\sqrt{2\pi(k-d)}} & x > 0 \\ 0 & x \leq 0 \end{cases} ,$$

and

$$f_{-|D|}(x) = \begin{cases} 0 & x > 0 \\ 2N(0, d) & x \leq 0 \end{cases} = \begin{cases} 0 & x > 0 \\ 2 \frac{e^{-\frac{x^2}{2d}}}{\sqrt{2\pi d}} & x \leq 0 \end{cases} .$$

Now we can calculate the desired probability

$$\begin{aligned} \Pr[z_{(1)} = z_{(2)}] &= \Pr[R_d > 0] \\ &= \int_0^\infty f_R(w) dw = \int_0^\infty f_{-|D|}(w) * f_{|S|}(w) dw \\ &= \int_0^\infty \frac{4}{2\pi\sqrt{d(k-d)}} \cdot \left[\int_{-\infty}^\infty e^{-\frac{x^2}{2(k-d)}} \cdot U(x) \cdot e^{-\frac{-(w-x)^2}{2d}} \cdot U(x-w) dx \right] dw \end{aligned}$$

⁸ This is a straightforward application of the Central Limit Theorem. It is also very easy to derive directly from the PDF of a normal random variable.

where $*$ denotes the convolution operator and $U(x)$ is the unit step function. By rearranging the terms we obtain

$$\begin{aligned} \Pr[z_{(1)} = z_{(2)}] &= \frac{4}{\sqrt{2\pi(k+1)}} \int_0^\infty \left(\frac{1}{\sqrt{2\pi\sigma^2}} \int_w^\infty e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \right) e^{-\frac{w^2}{2k}} dw \\ &= 2 \int_0^\infty \frac{1}{\sqrt{2\pi k}} [1 - \operatorname{erf}(\alpha w)] e^{-\frac{w^2}{2k}} dw \\ &= 2 \left[\int_0^\infty \frac{e^{-\frac{w^2}{2k}}}{\sqrt{2\pi k}} dw - \int_0^\infty \frac{e^{-\frac{w^2}{2k}} \cdot \operatorname{erf}(\alpha w)}{\sqrt{2\pi k}} dw \right] \\ &= 2 \left[\frac{1}{2} - \frac{1}{\pi} \arctan(\alpha\sqrt{2k}) \right] \end{aligned}$$

where $\sigma^2 = \frac{d(k-d)}{k}$, $\mu = \frac{\sigma^2 w}{d}$, $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ and $\alpha = \sqrt{\frac{d}{2k(k-d)}}$. The first of the two integrations is just a Gaussian over half of the space. As for the second integration this is a known definite integral [31]. Finally, substituting back with the original variables k and d we obtain

$$\Pr[z_{(1)} = z_{(2)}] = F_k(d) = 1 - \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{k-d}}\right).$$

□

Proof of Lemma 3

Proof. To show that the bits $\{z_{(i)}\}_{i=1}^n$ are uniform and independent, we need to show that the probability of $z_{(i)} = 0$ for any $i \in [1, n]$ is 0.5 and that the probability of $z_{(i)} = z_{(j)}$ for any $i, j \in [1, n]$ is 0.5. It is clear that when $y_{k+1} = 0$ the output of a PUF will be balanced. Therefore, it is straightforward to see that when $\{a_{(i)}\}_{i=1}^n$ are independent and chosen uniformly at random the bits $\{z_{(i)}\}_{i=1}^n$ will also be uniformly distributed. What remains to show is that there is no correlation between the bits $\{z_{(i)}\}_{i=1}^n$.

From Lemma 2 and as can be seen from Equation 3 the probability of any two PUF outputs being equal (or not equal) depends on the Hamming distance d between $P_{(i)} = Ua_{(i)}$ and $P_{(j)} = Ua_{(j)}$ and not the specific values of $a_{(i)}$ and $a_{(j)}$. Furthermore, we can deduce from Equation 3 that

$$\overline{F}_k(d) = \Pr[z_{(i)} \neq z_{(j)}] = 1 - F_k(d) = \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{k-d}}\right).$$

The two probability distributions $F_k(d)$ and $\overline{F}_k(d)$ are reflections of each other around $d = \frac{k}{2}$. Therefore, $\overline{F}_k(d) = F_k(k-d)$. This means that when the probability distribution of the Hamming distance between $P_{(i)}$ and $P_{(j)}$ is symmetrized around $\frac{k}{2}$, then $z_{(i)}$ and $z_{(j)}$ will be uncorrelated. To prove the rest of the lemma, we only need to show that the probability distribution of the Hamming distances between the different $P_{(i)}$ strings will be symmetric around $\frac{k}{2}$.

The n bit strings $\{a_{(i)}\}_{i=1}^n$ given to \mathcal{A} will induce $\frac{n(n-1)}{2}$ different Hamming distances between the corresponding $P_{(i)}$ strings. Recall from the lemma that $z_{(i)} = PUF_{Y_0, s_i}(a_{(i)})$. Therefore, as indicated by Equation 2 the PUF circuit will invert the $P_{(i)}$ strings based on the value of the bit s_i . From \mathcal{A} 's perspective s is chosen uniformly at random from $\{0, 1\}^n$. Therefore, each of the $P_{(i)}$ strings will be inverted with probability 0.5. For any two strings $P_{(i)}$ and $P_{(j)}$, if both of the strings or neither of them are inverted the Hamming distance d will not be affected. On the other hand, if only one of the two strings is inverted then the Hamming distance d will become $k - d$. Therefore, the Hamming distance between any two strings $P_{(i)}$ and $P_{(j)}$ will be d with probability 0.5 and will be $k - d$ with probability 0.5. This distribution is symmetric around $\frac{k}{2}$. \square

An Authentication Scheme Based on the Twisted Conjugacy Problem

Vladimir Shpilrain and Alexander Ushakov

¹ Department of Mathematics, The City College of New York, NY 10031, USA
`shpil@groups.sci.ccnycuny.edu`*

² Department of Mathematics, Stevens Institute of Technology,
Hoboken, NJ 07030, USA
`aushakov@stevens.edu`

Abstract. The conjugacy search problem in a group G is the problem of recovering an $x \in G$ from given $g \in G$ and $h = x^{-1}gx$. The alleged computational hardness of this problem in some groups was used in several recently suggested public key exchange protocols, including the one due to Anshel, Anshel, and Goldfeld, and the one due to Ko, Lee et al. Sibert, Dehornoy, and Girault used this problem in their authentication scheme, which was inspired by the Fiat-Shamir scheme involving repeating several times a three-pass challenge-response step.

In this paper, we offer an authentication scheme whose security is based on the apparent hardness of the *twisted conjugacy search problem* which is: given a pair of endomorphisms (i.e., homomorphisms into itself) φ, ψ of a group G and a pair of elements $w, t \in G$, find an element $s \in G$ such that $t = \psi(s^{-1})w\varphi(s)$ provided at least one such s exists. This problem appears to be very non-trivial even for free groups. We offer here another platform, namely, the *semigroup* of all 2×2 matrices over truncated one-variable polynomials over \mathbf{F}_2 , the field of two elements, with transposition used instead of inversion in the equality above.

1 Introduction

One of the most obvious ramifications of the discrete logarithm problem in the noncommutative situation is the *conjugacy search problem*:

Given a group G and two conjugate elements $g, h \in G$, find a particular element $x \in G$ such that $x^{-1}gx = h$.

This problem always has a recursive solution because one can recursively enumerate all conjugates of a given element, but this kind of solution can be extremely inefficient. Specific groups may or may not admit more efficient solutions, so the choice of the platform group is of paramount importance for security of a cryptographic primitive based on the conjugacy search problem. A great deal of research was (and still is) concerned with the complexity of this

* Research of the first author was partially supported by the NSF grant DMS-0405105.

problem in braid groups because there were several proposals, including the one by Anshel, Anshel, and Goldfeld [1], and the one by Ko, Lee et al. [11] on using the alleged computational hardness of this problem in braid groups to build a key exchange protocol. Also, Sibert, Dehornoy, and Girault [15] used this problem in their authentication scheme, which was inspired by the Fiat-Shamir scheme involving repeating several times a three-pass challenge-response step. At the time of this writing, no deterministic polynomial-time algorithm for solving the conjugacy search problem in braid groups has been reported yet; see [3] and [4] for recent progress in this direction. However, several heuristic algorithms, in particular so-called “length based attacks”, were shown to have very high success rates, see e.g. [7], [8], [10], [12], [13]. This shows that one has to be really careful when choosing the platform (semi)group to try to avoid length based or similar attacks. One way to achieve this goal is, informally speaking, to have “a lot of commutativity” inside otherwise non-commutative (semi)group; see [13] for a more detailed discussion.

In this paper, we propose an authentication scheme whose security is based on the apparent hardness of the *(double) twisted conjugacy search problem* which is:

given a pair of endomorphisms (i.e., homomorphisms into itself) φ, ψ of a group G and a pair of elements $w, t \in G$, find an element $s \in G$ such that $t = \psi(s^{-1})w\varphi(s)$ provided at least one such s exists.

This problem, to the best of our knowledge, has not been considered in group theory before, and neither was its decision version: given $\varphi, \psi \in \text{End}(G)$, $w, t \in G$, find out whether or not there is an element $s \in G$ such that $t = \psi(s^{-1})w\varphi(s)$. However, the following special case of this problem (called the *twisted conjugacy problem*) has recently attracted a lot of interest among group theorists:

given $\varphi \in \text{End}(G)$, $w, t \in G$, find out whether or not there is an element $s \in G$ such that $t = s^{-1}w\varphi(s)$.

This problem is very non-trivial even for free groups; see [5] for an astonishing solution in the special case where φ is an *automorphism* of a free group. To the best of our knowledge, this decision problem is open for free groups if φ is an arbitrary endomorphism. Another class of groups where the twisted conjugacy problem was considered is the class of polycyclic-by-finite groups [16]. Again, the problem was solved for these groups in the special case where φ is an automorphism.

The conjugacy problem is a special case of the twisted conjugacy problem, where φ is the identity map. Now a natural question is: what is the advantage of the more general (double) twisted conjugacy search problem over the conjugacy search problem in the context of an authentication scheme? The answer is: if the platform (semi)group G has “a lot” of endomorphisms, then Alice (the prover), who selects φ, ψ, w , and s , has an opportunity to select them in such a way that there are a lot of cancelations between $\psi(s), w$, and $\varphi(s)$, thus rendering length based attacks ineffective.

In this paper, we use the semigroup of all 2×2 matrices over truncated one-variable polynomials over \mathbf{F}_2 , the field of two elements, as the platform. It may seem that the platform necessarily has to be a group since one should at least have the element s (see above) invertible. However, as we will see in the next section, we do not really need the invertibility to make our authentication protocol work; what we need is just *some* antihomomorphism of G into itself, i.e., a map $*$: $G \rightarrow G$ such that $(ab)^* = b^*a^*$ for any $a, b \in G$. Every group has such an antihomomorphism; it takes every element to its inverse. Every *semigroup of square matrices* has such an antihomomorphism, too; it takes every matrix to its transpose. Some (semi)groups have other special antihomomorphisms; for example, any free (semi)group has an antihomomorphism that rewrites every element “backwards”, i.e., right-to-left. Here we prefer to focus on semigroups of matrices (over commutative rings) since we believe that these have several features making them fit to be platforms of various cryptographic protocols, see [14] for a more detailed discussion.

2 The Protocol

In this section, we give a description of a single round of our authentication protocol. As with the original Fiat-Shamir scheme, this protocol has to be repeated k times if one wants to reduce the probability of successful forgery to $\frac{1}{2^k}$.

Here Alice is the prover and Bob the verifier. Let G be the platform semigroup, and $*$ an antihomomorphism of G , i.e., $(ab)^* = b^*a^*$.

1. Alice’s public key is a pair of endomorphisms φ, ψ of the group G and two elements $w, t \in G$, such that $t = \psi(s^*)w\varphi(s)$, where $s \in G$ is her private key.
2. To begin authentication, Alice selects an element $r \in G$ and sends the element $u = \psi(r^*)t\varphi(r)$, called the *commitment*, to Bob.
3. Bob chooses a random bit c and sends it to Alice.
 - If $c = 0$, then Alice sends $v = r$ to Bob and Bob checks if the equality $u = \psi(v^*)t\varphi(v)$ is satisfied. If it is, then Bob accepts the authentication.
 - If $c = 1$, then Alice sends $v = sr$ to Bob and Bob checks if the equality $u = \psi(v^*)w\varphi(v)$ is satisfied. If it is, then Bob accepts the authentication.

Let us check now that everything works the way we want it to work.

- If $c = 0$, then $v = r$, so $\psi(v^*)t\varphi(v) = \psi(r^*)t\varphi(r) = u$.
- If $c = 1$, then $v = sr$, so $\psi(v^*)w\varphi(v) = \psi((sr)^*)w\varphi(sr) = \psi(r^*s^*)w\varphi(s)\varphi(r) = \psi(r^*)\psi(s^*)w\varphi(s)\varphi(r) = u$.

3 The Platform and Parameters

Our suggested platform semigroup G is the semigroup of all 2×2 matrices over truncated one-variable polynomials over \mathbf{F}_2 , the field of two elements. Truncated

(more precisely, N -truncated) one-variable polynomials over \mathbf{F}_2 are expressions of the form $\sum_{0 \leq i \leq N-1} a_i x^i$, where a_i are elements of \mathbf{F}_2 , and x is a variable. In

other words, N -truncated polynomials are elements of the factor algebra of the algebra $\mathbf{F}_2[x]$ of one-variable polynomials over \mathbf{F}_2 by the ideal generated by x^N .

Our semigroup G has a lot of endomorphisms induced by endomorphisms of the algebra of truncated polynomials. In fact, any map of the form $x \rightarrow p(x)$, where $p(x)$ is a truncated polynomial with zero constant term, can be extended to an endomorphism ϕ_p of the algebra of truncated polynomials. Indeed, it is sufficient to show that $\phi_p(x^N) = (p(x))^N$ belongs to the ideal generated by x^N , which is obviously the case if $p(x)$ has zero constant term. Then, since ϕ_p is both an additive and a multiplicative homomorphism, it extends to an endomorphism of the semigroup of all 2×2 matrices over truncated one-variable polynomials in the natural way.

If we now let the antihomomorphism $*$ from the description of the protocol in our Section 2 to be the matrix transposition, we have everything set up for an authentication scheme using the semigroup G as the platform.

Now we have to specify parameters involved in our scheme. The parameter N determines the size of the key space. If N is on the order of 300, then there are 2^{300} polynomials of degree $< N$ over \mathbf{F}_2 , so there are 2^{1200} 2×2 matrices over N -truncated polynomials, i.e., the size of the private key space is 2^{1200} , which is large enough.

At the same time, computations with (truncated) polynomials over \mathbf{F}_2 are very efficient (see e.g. [2], [6], or [9] for details). In particular,

- Addition of two polynomials of degree N can be performed in $O(N)$ time.
- Multiplication of two polynomials of degree N can be performed in $O(N \log_2 N)$ time.
- Computing composition $p(q(x)) \bmod x^N$ of two polynomial of degree N can be performed in $O((N \log_2 N)^{\frac{3}{2}})$ time (see e.g. [6, p.51]).

Since those are the only operations used in our protocol, the time complexity of executing a single round of the protocol is $O((N \log_2 N)^{\frac{3}{2}})$.

The size of public key space is large, too. One public key is, again, a 2×2 matrix over N -truncated polynomials, and two other public keys are endomorphisms of the form $x \rightarrow p(x)$, where $p(x)$ is an N -truncated polynomial with zero constant term. Thus, the number of different endomorphisms in this context is on the order of 2^{300} , hence the number of different *pairs* of endomorphisms is on the order of 2^{600} .

We also have to say a few words about how a private key $s \in G$ is selected. We suggest that all entries of the matrix s have non-zero constant term; other coefficients of the entries can be selected randomly, i.e., “0” and “1” are selected with probability $\frac{1}{2}$ each. Non-zero constant terms are useful here to ensure that there are sufficiently many non-zero terms in the final product $t = \psi(s^*)w\varphi(s)$.

4 Cryptanalysis

As we have pointed out in the previous section, the key space with suggested parameters is quite large, so that a “brute force” attack by exhausting the key space is not feasible.

The next natural attack that comes to mind is attempting to solve a system of equations over \mathbf{F}_2 that arises from equating coefficients at the same powers of x on both sides of the equation $t = \psi(s^*)w\varphi(s)$. Recall that in this equation t, w, φ , and ψ are known, whereas s is unknown.

More specifically, our experiments emulating this attack were designed as follows. The entries of the private matrix s were generated as polynomials of degree $N-1$, with $N = 100$ (which is much smaller than the suggested $N = 300$), with randomly selected binary coefficients, except that the constant term in all polynomials was 1. Then, the endomorphisms φ and ψ were of the form $x \rightarrow p_i(x)$, where $p_i(x)$ are polynomials of degree $N-1$, with $N = 150$, with randomly selected binary coefficients, except that the constant term in both of them was 0. Finally, the entries of the public matrix w were generated, again, as polynomials of degree $N-1$, with $N = 100$, with randomly selected binary coefficients, except that the constant term in all polynomials was 1.

The attack itself then proceeds as follows. The matrix equation $t = \psi(s^*)w\varphi(s)$ is converted to a system of $4N$ polynomial equations (N for each entry of a 2×2 matrix) over \mathbf{F}_2 . The unknowns in this system are coefficients of the polynomials of degree $N-1$ that are the entries of the private matrix s . Then, starting with the constant term and going up, we equate coefficients at the same powers of x on both sides of each equation. After that, again starting with the coefficients at the constant term and going up, we find all possible solutions of each equation, one at a time. Thus we are getting a “tree” of solutions because some of the unknowns that occur in coefficients at lower powers of x also occur in coefficients at higher powers of x . If this tree does not grow too fast, then there is a chance that we can get all the way to the coefficients at highest power of x , thereby finding a solution of the system. This solution may not necessarily yield the same matrix s that was selected by Alice, but it is sufficient for forgery anyway.

We have run over 1000 experiments of this kind (which took about two weeks), allowing the solution tree to grow up to the width of 16384, i.e., allowing to go over at most 16384 solutions of each equation when proceeding to a higher power of x . Each experiment ran on a personal computer with Pentium 2Ghz dual core processor. The success rate of the described attack with these parameters was 0%.

5 Conclusions

We have introduced:

1. An authentication scheme based on the (double) twisted conjugacy problem, a new problem, which is allegedly hard in some (semi)groups.
2. A new platform semigroup, namely the semigroup of all 2×2 matrices over truncated one-variable polynomials over \mathbf{F}_2 . Computation in this semigroup

is very efficient and, at the same time, the non-commutative structure of this semigroup provides for security at least against obvious attacks.

We point out here one important advantage of using the (double) twisted conjugacy problem over using a more “traditional” conjugacy search problem as far as (semi)groups of matrices are concerned. The conjugacy search problem admits a linear algebra attack upon rewriting the equation $x^{-1}gx = h$ as $gx = xh$; the latter translates into a system of n^2 linear equations with n^2 unknowns, where n is the size of the matrices involved, and the unknowns are the entries of the matrix x . Of course, if the entries come not from a field but from a more general ring, such a system of linear equations does not necessarily admit a straightforward solution, but methods emulating standard techniques (like Gauss elimination) usually have a pretty good success rate anyway. For the twisted conjugacy problem, however, there is no reduction to a system of linear equations.

We have considered an attack based on reducing the twisted conjugacy problem to a system of *polynomial* equations over \mathbf{F}_2 , but this attack becomes computationally infeasible even with a much smaller crucial parameter (which is the maximum degree of the polynomials involved) than the one we suggest in this paper.

References

1. Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. *Math. Res. Lett.* 6, 287–291 (1999)
2. Bini, D., Pan, V.: *Polynomial and Matrix Computations. vol. 1: Fundamental Algorithms.* Birkhäuser, Basel (1994)
3. Birman, J., Gebhardt, V., Gonzalez-Meneses, J.: Conjugacy in Garside groups I: Cyclings, powers, and rigidity. *Groups, Geometry and Dynamics* 1, 221–279 (2007)
4. Birman, J., Gebhardt, V., Gonzalez-Meneses, J.: Conjugacy in Garside groups II: Structure of the ultra summit set. *Groups, Geometry and Dynamics* 2, 13–61 (2008)
5. Bogopolski, O., Martino, A., Maslakova, O., Ventura, E.: Free-by-cyclic groups have solvable conjugacy problem. *Bull. London Math. Soc.* 38, 787–794 (2006)
6. Bürgisser, P., Clausen, M., Shokrollahi, M.A., Lickteig, T.: *Algebraic Complexity Theory.* Springer, Heidelberg (1997)
7. Garber, D., Kaplan, S., Teicher, M., Tsaban, B., Vishne, U.: Length-based conjugacy search in the Braid group. *Contemp. Math., Amer. Math. Soc.* 418, 75–87 (2006)
8. Garber, D., Kaplan, S., Teicher, M., Tsaban, B., Vishne, U.: Probabilistic solutions of equations in the braid group. *Advances in Applied Mathematics* 35, 323–334 (2005)
9. von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*, 2nd edn. Cambridge University Press, Cambridge (2003)
10. Hofheinz, D., Steinwandt, R.: A practical attack on some braid group based cryptographic primitives. In: Desmedt, Y.G. (ed.) *PKC 2003. LNCS*, vol. 2567, pp. 187–198. Springer, Heidelberg (2002)
11. Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J., Park, C.: New public-key cryptosystem using braid groups. In: Bellare, M. (ed.) *CRYPTO 2000. LNCS*, vol. 1880, pp. 166–183. Springer, Heidelberg (2000)

12. Myasnikov, A.D., Ushakov, A.: Length based attack in braid groups. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 76–88. Springer, Heidelberg (2007)
13. Ruinskiy, D., Shamir, A., Tsaban, B.: Cryptanalysis of group-based key agreement protocols using subgroup distance functions. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 61–75. Springer, Heidelberg (2007)
14. Shpilrain, V.: Hashing with polynomials. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 22–28. Springer, Heidelberg (2006)
15. Sibert, H., Dehornoy, P., Girault, M.: Entity authentication schemes using braid word reduction. *Discrete Applied Math.* 154-2, 420–436 (2006)
16. Fel'shtyn, A., Troitsky, E.: Twisted conjugacy separable groups, preprint, <http://arxiv.org/abs/math/0606764>

Restricted Queries over an Encrypted Index with Applications to Regulatory Compliance

Nikita Borisov¹ and Soumyadeb Mitra²

¹ Department of Electric and Computer Engineering
University of Illinois at Urbana–Champaign

nikita@uiuc.edu

² Data Domain

soumyadeb@gmail.com

Abstract. Compliance storage is an increasingly important area for businesses faced with a myriad of new document retention regulations. Today, businesses have turned to Write-One Read Many (WORM) storage technology to achieve compliance. But WORM answers only a part of the compliance puzzle; in addition to guaranteed document retention, businesses also need *secure indexing*, to ensure auditors can find required documents in a large database, *secure deletion* to expire documents (and their index entries) from storage once they are past their expiry period, and support for *litigation holds*, which require that certain documents are retained pending the resolution of active litigation.

We build upon previous work in compliance storage and attribute-based encryption to design a system that satisfies all three of these requirements. In particular, we design a new encrypted index, which allows the owner of a database of documents to grant access to only those documents that match a particular query. This enables litigation holds for expired documents, and at the same time restricts auditor access for unexpired documents, greatly limiting the potential for auditor abuse as compared to previous work. We show by way of formal security proofs that our construction is secure and that it prevents reconstruction attacks wherein the index is used to recover the contents of the document. Our experiments show that our scheme can be practical for large databases and moderate sizes of queries.

1 Introduction

Recent regulations require many corporations to ensure trustworthy long-term retention of their routine business documents. The US alone has over 10,000 regulations that mandate how business data should be managed over their entire lifecycle [1]. The focus of most of these regulations (e.g., SEC Rule 17a 4 [2], Heath Insurance Portability and Accountability Act, 1996 (HIPAA), and the Sarbanes–Oxley Act, 2002 [3]). is to ensure that the records are kept immutable during their retention periods and are securely erased at the end of their lifecycle. For example, the HIPAA Section 1173(d) sets security standards for health information, including safeguards to ensure the integrity and confidentiality of

the information and to protect against any reasonably anticipated threats or hazards to the integrity of the information once stored.

This has led to a rush to introduce write-once read-many (WORM) compliance storage devices (e.g., [4,5,6]). These devices ensure *term immutability* of files. Every file committed to the device has an expiry date either assigned explicitly by the committing application or as a system default. Until the end of its expiration period, the file is read-only and cannot be deleted or altered even by a superuser. A WORM device hence secures critical documents from certain threats originating from company insiders or hackers with administrative privileges. Once a file expires, the device lets the file be deleted, so that external cleanup applications can erase it. Cleanup is required as expired records can often be a liability for the company—e.g., it can be subpoenaed in future lawsuits or regulatory enquiries.

Records are often subject to litigation holds. A record subject to a litigation hold must not be erased from the device until the hold is removed. Litigation holds can be enforced by disabling the automatic deletion (on expiry) of the files, an operation supported by most WORM devices.

Unfortunately, this is not enough to ensure the trustworthiness of litigation holds. Litigation hold requests are often expressed as keyword queries. For example, the judge or auditor might require all email having the words “Martha and Ralph and ImClone” to be retained. Furthermore, the person invoking the hold (judge/auditor) often does not have direct access to the WORM device when the hold request is being enforced—the hold is executed by a company employee (e.g. sysadmin) who may not be trusted in our threat model. In such a scenario, the judge or auditor must be able to verify (at a future point in time) that all the files that are subject to the litigation hold query have been retained.

A simple way to address this problem is to retain the inverted index created over the documents [7] on the WORM device even after the documents have expired. At any future point, the index can be used by the auditor/judge to rerun the litigation hold query to verify that all the resulting documents have been retained. Unfortunately, a keyword index contains too much sensitive information, as the index entries can be used to reconstruct the complete set of keywords in every document, even those which are not subject to the litigation hold.

To address this problem, we design a new type of encrypted index that allows the owner to create a key that gives an auditor access to only those documents that match a particular query. Further, the index prevents reconstruction attacks for any documents that do not match the query. The integrity of the key can be verified, so an auditor checking a litigation hold can be sure that no documents have been removed. Additionally, these per-query keys can be used to restrict auditor access in situations not involving litigation holds, so that auditor access to a WORM device can be restricted to only the documents related to the matter at hand, reducing the potential for auditor abuse.

We implement our scheme and evaluate it using the Enron data set [8], representative of a database at a large enterprise. We show that, although the

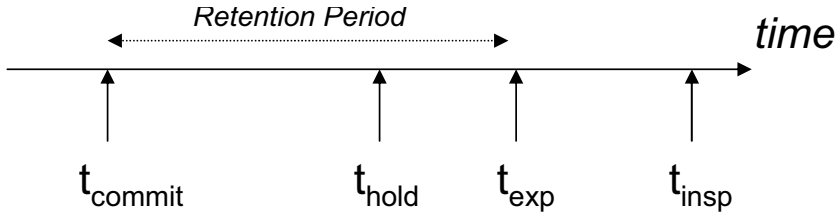


Fig. 1. Record Lifecycle

encrypted index is both storage- and CPU-intensive, it can be practical for large database sizes and moderate query lengths.

2 Background

2.1 Threat Model for Compliance Records

We use the term *record* to refer to business documents with a fixed retention period, such as financial notes, email, memos, reports, and instant messages. Figure 1 shows the life cycle of record in a typical business environment.

At time t_{commit} a user “Alice” creates a record and commits it to the WORM storage server. We assume that the commit is trustworthy, that is, Alice properly stores the document and correctly sets the expiry time to t_{exp} .

A user Bob (e.g. a judge) subjects the record to litigation hold at time t_{hold} where ($t_{\text{commit}} \leq t_{\text{hold}} \leq t_{\text{exp}}$). In this paper, we consider litigation hold requests that are expressed as keyword queries. An example hold request would be:

All documents containing “Martha” and “Ralph” that were created between 06/2002 and 08/2002 must be retained.

We assume that the user Bob does not have direct access to the WORM device for enforcing the hold request. The hold is implemented by a company employee Mala (e.g. the system administrator)—she must run the litigation hold query to obtain a list of matching documents and ensure that the documents are not deleted on expiry (by disabling their auto-deletion) until they have been inspected by Bob at time t_{insp} .

We however do not trust the user Mala to enforce the hold request properly. For example, Mala might be the CEO who retroactively wants to hide an illegal email conversation she had with her broker about whether to sell stock in her company. Mala cannot alter the email itself, because it is on WORM storage. Mala can however tamper with the litigation hold process. For example, out of all the files that satisfy the litigation hold query, she might retain only the “safe” files while expire the sensitive ones.

At the time of inspection (t_{insp}) Bob must be able to verify the completeness of the litigation hold query result. That is, he should be able to provably verify that all the files satisfying his hold query have been retained. The trivial way

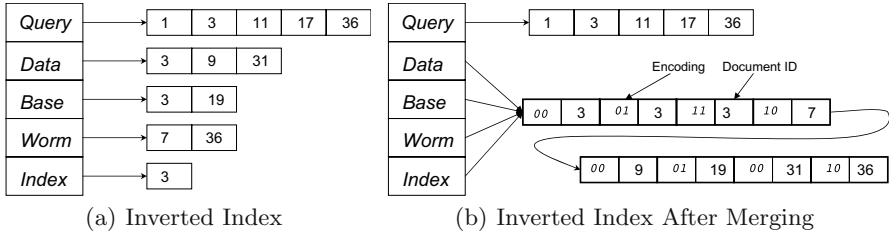


Fig. 2. Posting Lists. With each keyword, a *posting list* of documents containing that keyword is stored. After merging, the keyword (or its hash) must also be stored in the posting list.

of ensuring this is to retain the inverted index even after the documents have expired. Bob can run the litigation hold query and obtain the complete list of documents satisfying the query—all those documents must have been retained by Mala.

Unfortunately, the user Bob, although law-abiding, is inquisitive—he may try to reconstruct or extract information about other expired records which do not satisfy his hold request. If the entire inverted index is retained as described above, he can extract information about such documents from the index. Hence, such a scheme is undesirable. Our goal is to develop an index structure which would let Bob obtain the list of documents which satisfy his query without allowing him learn any information about other documents.

2.2 Storage Model

In this paper, we consider a WORM device with a file system interface [6], though our techniques are equally applicable to object-based devices [4]. The interface allows users to create new files and to append to existing files. Appends are required for indexing and can be efficiently supported since the underlying media are magnetic [7]. The append feature should be restricted to the specific storage volume holding the index, to prevent appends to committed files that contain ordinary records.

2.3 Querying and Indexing

The standard query interface for semi-structured and unstructured business records is keyword queries, where a user types an arbitrary set of terms and obtains a list of the documents containing some or all of the terms. Queries can be further constrained by a document creation time interval.

The standard implementation of keyword queries uses an inverted index [9]. As shown in Figure 2.3(a), an inverted index consists of a dictionary of terms and a *posting list* of the identifiers (IDs) of the documents that contain that term. In addition to an ID, each posting list element gives the number of occurrences of that term in the document (its *term frequency*), which is not shown in the figure.

The document IDs are usually assigned in order of document arrival, through an increasing counter. Queries are answered by scanning the posting lists of the terms in the query, thereby obtaining a list of documents having some/all of the keywords. The resulting documents are ranked based on the number of occurrences of the keywords and their relative importance [9].

As explained earlier, we assume that litigation hold queries are also expressed as keyword queries.

2.4 Compliant Inverted Indexes

Because the volume of compliance data is so large and no one wants to wait hours or days for a query answer, index lookup is the only practical record search method. But this means that an adversary can make a record inaccessible by omitting its entries from the index, or altering the index to point to a different version of the record on WORM. Hence, a record can be *logically deleted* or *logically modified* by suitably altering the index structure. To prevent such tampering, the index must be kept on WORM media [7,10].

An inverted index can be stored on WORM by keeping each posting list in an append-only WORM file. The index can be updated when a new document is added, by appending its document ID to the posting lists of all the keywords it contains. Unfortunately, this operation can prohibitively slow as each append would incur a random I/O. Mitra *et al* have studied the problem of efficiently updating an inverted index on WORM [7] by merging posting lists (into a maximum of as many lists as the number of cache blocks in the storage server [7]). Each merged list contains the union of the document IDs from the individual posting lists that are merged together. A keyword encoding is also stored in each posting element, to identify which keyword in the merged set appears in that document.

The litigation hold techniques that we have developed here treat each keyword posting list as a separate logical unit independent of whether it is stored merged with other keyword posting lists. Specifically, the document IDs are stored encrypted with a keyword specific key. The querier hence would be able to decrypt only those document IDs (in the merged list) which satisfy the query though he would have to run decryption on all the posting elements. Hence, in our case there is no need to store the keyword encoding separately with each posting element.

A compliant index structure must also support deletion. An inverted index contains sufficient information to *reconstruct* the set of indexed terms in a document. For example, from the index in Figure 2.3(a) Bob can learn that document 3 contains the words Query, Data, Base and Index. From the set of words in a document, an adversary can often infer its meaning (e.g., “fire next Thomas week”). Thus it is critical to clean up the index when documents are deleted.

The easiest way to support deletion is to divide the documents into groups called *disposition group* based on their commit times (e.g. all documents committed within a week form one group) and to create a separate index for every disposition group, as shown in Figure 3. The expiry time of a posting list file is

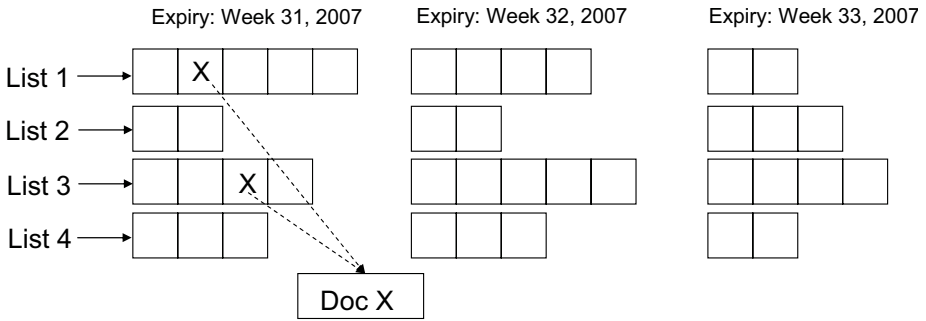


Fig. 3. The split index provides deletion by creating a separate index for each disposition group

set to that of the corresponding disposition group. Once the disposition group expires, the posting list files are deleted. This deletion scheme is strongly secure: after the posting lists are deleted, the adversary cannot get any information about the deleted documents.

Unfortunately, this approach has very poor query performance. A single-keyword query requires scanning as many posting list files as the number of disposition groups spanned by the query interval. Mitra *et al* have proposed more efficient techniques for supporting secure deletion with better query performance by encrypting the posting list elements with a disposition group specific key and adding noise terms [11]. Our scheme builds on top of this design by replacing the encryption of posting list elements with a new scheme, described in Section 4.

3 Litigation Hold Approaches

In the rest of the paper, we consider a litigation hold request of the following form:

Retain all documents which satisfy the query Q and were committed in disposition interval \mathcal{D}

where Q is an arbitrary boolean keyword query involving conjunction of keywords. (A disjunctive query can be implemented as a collection of several separate litigation holds.) All the documents committed in the disposition group \mathcal{D} must be retained in response to this hold request. (although we consider a single disposition interval, the techniques are trivially extendible to multiple disposition intervals).

As shown in Figure 1, Bob invokes the hold request at time t_{hold} . However, the verification of whether all the documents subject to the hold have been retained is done at a later point in time at t_{insp} , where t_{insp} could be after the expiry

time of disposition group \mathcal{D} . Bob must be able to verify that all the documents in \mathcal{D} satisfying his hold query Q have been retained.

3.1 Trivial Approach

The trivial approach to this problem is to require Mala to retain the entire inverted index for the documents in \mathcal{D} till t_{insp} , along with the documents which satisfy query Q . At t_{insp} . Bob can rerun Q on the index and get a list of documents satisfying Q . These are the documents which should have been retained by Mala.

Since the index is on WORM, Mala cannot tamper with the index. When the index expires, she can delete it and create a new tampered copy of the index. However, Bob will be able to detect this attack by checking the *create-time* of the index.

Unfortunately, this approach also lets Bob get information about documents which do not satisfy his query. An inverted index contains sufficient information to *reconstruct* the set of indexed terms in a document. For example, from the index in Figure 1(a) Bob can learn that document 3 contains the words Query, Data, Base and Index. From the set of words in a document, an adversary can often infer its meaning (e.g., “fire Harriet tomorrow”).

This problem can be partially addressed by retaining only the posting lists of keywords involved in the hold query Q . This prevents Bob from reconstructing the contents of arbitrary documents. However, Bob can still learn about documents which do not satisfy his query. For example, even if no documents in \mathcal{D} match the query “Martha and Ralph”, Bob may learn that a large number of documents contained the keyword “Martha” during that disposition and infer sensitive information. (Normally, noise techniques proposed by Mitra *et al* [11] prevent such leaks).

Furthermore, a large company may be involved in many separate pieces of litigation and thus be subject to many holds. An auditor involved with one litigation will have access to all of the documents corresponding to all other litigation, which is undesirable.

3.2 Key-Policy Attribute-Based Encryption

A possible way to resolve this problem is to use Key-Policy Attribute-Based Encryption (KP-ABE) [12]. KP-ABE allows one to encrypt a document together with a set of attributes, and then to create a decryption key that allows the decryption of only those documents that satisfy a particular access structure over a document. An access structure can be thought of as a subset of $2^{\mathcal{A}}$, where \mathcal{A} is a set of attributes; documents whose attribute set is in the structure are said to satisfy the structure. As a simple example, a query such as $A \wedge B$ can define an access structure of all attributes sets S such that $\{A, B\} \subset S$.

Consider how an KP-ABE system would be used given attributes A, B, C, D and documents M_1, M_2 . Suppose the two documents were encrypted as follows: $E_1 = \text{Encrypt}(M_1, \{A, B, C\})$ and $E_2 = \text{Encrypt}(M_2, \{A, C, D\})$. Then a key

K_1 corresponding to the access structure $A \wedge B$ could be used to decrypt E_1 but not E_2 . But K_2 , corresponding to the access structure $A \wedge (B \vee C)$ could be used to decrypt both documents.

Mapping KP-ABE to our setting, each potential keyword could be assigned to be an attribute. Each index entry for a keyword K occurring in document D would then be encrypted under the set of keywords contained in D . Bob would then be given a key corresponding to the litigation hold query, and he would be able to decrypt only those index entries corresponding to documents matching his query. Other index entries would be indistinguishable from noise terms. In addition, all document contents can be encrypted with KP-ABE using the set of keywords that occur in the document. This would further ensure that Bob could not read any document that does not match his litigation hold, both among the expired documents and those that are currently still within their retention period. As such, Bob would be prevented from abusing his access to the WORM drive to learn company trade secrets or other sensitive information.

However, this scheme has a fatal flaw. A document encrypted under KP-ABE includes, in cleartext, the list of all the attributes that it is encrypted under. This would allow Bob to easily reconstruct document contents from encrypted index entries (or encrypted documents). Thus the situation is even worse than with the trivial approach, since the KP-ABE encryption allows Bob to recover all the keywords in the non-matching documents, rather than only those that are a part of the litigation hold query.

We therefore design a scheme that is based on KP-ABE, but addresses this flaw. It has the advantage that Bob cannot read any document that is not covered by his query, but it also prevents the reconstruction attacks possible with KP-ABE.

4 Encrypted Index with Per-query Keys

4.1 Preliminaries

Our scheme, as with KP-ABE, is based on using bilinear maps:

Definition 1 (Bilinear Map). *Let G_1, G_2, G_T be cyclic groups of same prime order p , and let g_1 and g_2 be generators of G_1 and G_2 respectively. Then $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear map if:*

1. For any a, b, x, y , $e(a^x, b^y) = e(a, b)^{xy}$ (bilinearity)
2. $e(g_1, g_2) \neq 1$ (non-degeneracy)

Efficiently-computable bilinear maps are created based on Weil and Tate pairings over elliptic curves [13, 14]. Usually, $G_1 = G_2$ (or equivalently, there is an efficiently-computable isomorphism between G_1 and G_2 and vice versa.) However, we will use maps where $G_1 \neq G_2$, because we need the following additional assumption:

Assumption 1 (External Diffie–Hellman (XDH)). *Given groups G_1, G_2, G_T and a bilinear map e between them, the External Diffie–Hellman assumption holds if the Decisional Diffie–Hellman problem is hard in G_1 . In other words, there is no polynomial time algorithm that can decide whether, given $g_1^a, g_1^b, g_1^c, c = ab$ with a non-negligible advantage.*

It is easy to see that if $G_1 = G_2$, DDH is easy to solve by computing $e(g^a, g^b)$ and $e(g^c, g)$. However, in bilinear maps based on MNT curves [15], where $G_1 \neq G_2$, the XDH assumption is believed to hold [16].

We also require a variant of the standard Bilinear Diffie–Hellman (BDH) assumption [17] for pairings where $G_1 \neq G_2$:

Definition 2 (Asymmetric Bilinear Diffie–Hellman (ABDH)). *Given groups G_1, G_2, G_T , with generators g_1 of G_1 and g_2 of G_2 and a bilinear map e between them, the Asymmetric Bilinear Diffie–Hellman Assumption holds if there is no efficient algorithm that, given as input g_2^a, g_2^b, g_2^c , computes $e(g_1, g_2)^{abc}$.*

Note that due to the existence of an efficiently-computable isomorphism from G_2 to G_1 , this assumption is weaker than a variant that uses g_1 rather than g_2 .

4.2 Set Up

We consider a collection of documents $\mathcal{D} = \{D_1, \dots, D_n\}$ and a collection of keywords \mathcal{K} . For ease of exposition, we will assume that $\mathcal{K} \subset \mathbb{N}$, since the actual names of the keywords do not matter for our purposes. Let $k : \mathcal{D} \rightarrow 2^{\mathcal{K}}$ be the function relating documents to keywords.

In our scheme, each document D_i will be stored under a secret document identifier, id_i . In addition, each keyword will have a corresponding index of encrypted document identifiers. In practice, the WORM drive can contain a hash table mapping identifiers id_i to file names. The secret identifier can also be used to derive an encryption key for the document, making it impossible to read a document without knowing its identifier.

For the purposes of encryption, Alice must generate a secret value s , as well as secrets s_1, \dots, s_m corresponding to each keyword. These values can be computed from a master secret as, e.g., an HMAC [18] of the keyword name. New secrets will be computed for every disposition period.

Alice will also commit to WORM storage $g_1^{s_1}, \dots, g_1^{s_m}$ and $e(g_1, g_2)^s$. These will be used to verify the integrity of the per-query keys.

4.3 Storage

To store a document D_i , the Alice picks a random identifier d_i . Then, for each keyword $j \in k(D_i)$, Alice adds $g_1^{d_i s_j}$ to the index I_j . Finally, the document identifier is calculated as $id_i = H(e(g_1, g_2)^{s d_i})$, where H is a hash function.

4.4 Lookup by Alice

Alice can easily use the indices to look up a document. For example, to look up a document that has keywords k_1 and k_2 , Alice first reads all the identifiers in index I_{k_1} and raises them to the power $s_{k_1}^{-1}$:

$$\left(g_1^{d_i s_{k_1}}\right)^{s_{k_1}^{-1}} = g_1^{d_i}$$

She then performs the same transform on the index I_{k_2} and looks for matches. Given $g_1^{d_i}$ that occurs in both sets, the document ID can easily be computed as $H(e(g_1^{d_i}, g_2)^s)$.

4.5 Per-query Key

Alice can also generate a key that can be used by Bob without knowing the per-keyword secrets. This key will allow the lookup of documents that match *all* the keywords from a selected set. Given keywords k_1, k_2, \dots, k_l , Alice picks a_1, \dots, a_l , such that:

$$s_{k_1} a_1 + s_{k_2} a_2 + \dots + s_{k_l} a_l \equiv s \pmod{p}$$

It is easy to see that a_1, \dots, a_{l-1} can be picked randomly, with a_l being used to solve the remaining equation. Alice then gives $g_2^{a_1}, g_2^{a_2}, \dots, g_2^{a_l}$ to Bob. Bob can verify that they key is constructed correctly based on the committed values:

$$\pi_{i=1}^l e(g_1^{s_{k_i}}, g_2^{a_i}) = e(g_1, g_2)^{\sum_{i=1}^l s_{k_i} a_i} = e(g_1, g_2)^s$$

4.6 Lookup Using a Per-query Key

If Bob is in possession of a key corresponding to keywords k_1, \dots, k_l , he can find documents matching the conjunction of these keywords as follows. First, for each entry $g_1^{d_i s_{k_1}}$ in index I_{k_1} , Bob computes the pairing:

$$e(g_1^{d_i s_{k_1}}, g_2^{a_1}) = e(g_1, g_2)^{d_i s_{k_1} a_1}$$

Similar pairings are computed for each entry in indices I_{k_2}, \dots, I_{k_l} . Then for each tuple of documents $(e(g_1, g_2)^{d_{i_1} s_{k_1} a_1}, \dots, e(g_1, g_2)^{d_{i_l} s_{k_l} a_l})$, Bob computes the product:

$$\prod_{j=1}^l e(g_1, g_2)^{d_{i_j} s_{k_j} a_j}$$

If the entries in the tuple all correspond to the same document d_i , then the result will be:

$$\begin{aligned} \prod_{j=1}^l e(g_1, g_2)^{d_i s_{k_j} a_j} &= \\ e(g_1, g_2)^{(\sum_{j=1}^l s_{k_j} a_j) d_i} &= e(g_1, g_2)^{d_i s} \end{aligned}$$

Which will make it possible to look up the document. If the d_i 's are not equal, the result will be random, and the next tuple should be considered, until all of $|I_1| \cdot |I_2| \cdots |I_l|$ tuples have been checked.

5 Proof of Security

In our scheme, we are concerned with two security properties. First, we want to ensure that the auditor Bob is not able to read any expired document that does not match the particular litigation hold query. Second, we want to prevent the reconstruction attack, which entails hiding the existence of documents that are not matched by a litigation hold.

5.1 Document Secrecy

To ensure document secrecy, the adversary must be unable to determine $e(g_1, g_2)^{ds}$ for a document d for which he does not have a key. We set up the document secrecy game as follow:

Definition 3 (Document Secrecy Game)

Parameters: *The game is parameterized by groups G_1, G_2, G_T , generators g_1, g_2 , bilinear map e and keyword set \mathcal{K} .*

Keyword selection: *The adversary picks a set of keywords $\gamma \subset \mathcal{K}$ that he wishes to be challenged upon.*

Setup: *The challenger picks secret parameters s, s_i and commits the public parameters $g_2^{s_i}, e(g_1, g_2)^s$ to the adversary.*

Challenge: *The challenger picks a document id d and provides $g_1^{ds_i}$ for $i \in \gamma$.*

Learning phase: *The adversary submits to the challenger a polynomial number of queries $\gamma' \subset \mathcal{K}$ such that $\gamma' \not\subset \gamma$. For each γ' , the challenger returns the corresponding decryption key $\{g_2^{a_i}\}_{i \in \gamma'}$ such that $\sum_{i \in \gamma'} a_i s_i = s$.*

Response: *The adversary outputs ω , which is his best guess for $e(g_1, g_2)^{sd}$.*

In essence, the game requests that an adversary, given index entries for a document that has keywords γ , must find out the corresponding document ID, while being able to obtain decryption keys for any set of keywords, as long as one of them is not in γ , and hence the keys should not allow the adversary to decrypt the document ID. The advantage of an adversary is defined as:

$$Pr(\omega = e(g_1, g_2)^{sd}) - \frac{1}{|G_T|}$$

Theorem 1 (Document Secrecy)

For any polynomial-time adversary A that has advantage ϵ at the Document Secrecy Game, there is a polynomial-time adversary A' that has advantage ϵ at the Asymmetric Bilinear Diffie–Hellman Game with the same groups.

The proof of this theorem is modeled upon the proof of security of KP-ABE [12], due to the similarities between the two schemes.

Proof. We build a simulator A' that will solve the ABDH problem given A as input. Recall that in the ABDH problem, A' will be given (g_2^a, g_2^b, g_2^c) and A' must compute $e(g_1, g_2)^{abc}$.

A' starts by defining a set of keywords \mathcal{K} and running A to obtain the challenge keyword set γ . A' now defines the secret parameters as follows: It picks $r_i \in_R \mathbb{Z}_p$ for all $i \in \mathcal{K}$. It then defines $s_i = r_i$ for $i \in \gamma$ and $s_i = br_i$ for all $i \notin \gamma$. It also sets $s = ab$.

A' outputs the public parameters: $g_2^{s_i} = g_2^{r_i}$ for $i \in \gamma$, $g_2^{s_i} = (g_2^b)^{r_i}$ for $i \notin \gamma$, and $e(g_1, g_2)^s = e(g_1^a, g_2^b)$. (g_1^a can be obtained using the isomorphism ϕ from G_2 to G_1 .)

A' then outputs the challenge document, with $d = c$:

$$\phi(g_2^c)^{r_i} = g_1^{cr_i} = g_1^{ds_i}$$

A' then runs A , which will proceed to issue queries of the form γ' . Let $\gamma' = \{k_1, \dots, k_n\}$, and assume without loss of generality that $k_1 \notin \gamma$. A' needs to produce $g_2^{a_1}, \dots, g_2^{a_n}$ such that $\sum_{i=1}^n a_i s_i = s$.

We first compute $g_2^{a'_1}, \dots, g_2^{a'_n}$ such that $\sum_{i=1}^n a'_i r_i = a$. We do this by choosing a'_2, \dots, a'_n randomly and computing:

$$g_2^{a'_1} = \left(\frac{g_2^a}{\prod_{i=2}^n g_2^{a'_i r_i}} \right)^{r_1^{-1}}$$

For $i \in \gamma$, we set $a_i = a'_i b$, and compute $g_2^{a_i} = (g_2^b)^{a'_i}$. (Since $i \neq 1$, a'_i is chosen by A' .) For $i \notin \gamma$, we set $a_i = a'_i$. In both cases, we have that $a_i s_i = (a'_i r_i) b$. Therefore, $\sum_{i=1}^n a_i s_i = ab$.

Finally, A will output ω as a guess for $e(g_1, g_2)^{ds}$. If the guess is correct, $\omega = e(g_1, g_2)^{abc}$ by construction. Therefore, when A' outputs ω , it will enjoy the same advantage as A .

5.2 Reconstruction Attack

To prevent the reconstruction attack, we must ensure that an adversary does not learn anything from the index, other than the identifiers of those documents he should have access to. The noise terms added using the scheme from [11] can be used to ensure that the size of each posting list reveals no information to the attacker. In addition, we want to ensure that given two entries in two posting lists, an attacker should be unable to discover whether those entries correspond to the same or different documents. We therefore set up the reconstruction game as follows.

Definition 4 (Reconstruction Game).

Parameters: *The game is parameterized by $G_1, G_2, G_T, g_1, g_2, e,$ and \mathcal{K} .*

Keyword selection: *The adversary picks two keywords $k_1, k_2 \in \mathcal{K}$ that he wishes to be challenged upon.*

Setup: *The challenger defines the private parameters s_i, s and the corresponding public parameters $g_2^{s_i}$ and $e(g_1, g_2)^s$.*

Challenge: *The challenger picks two document IDs, d_0 and d_1 and flips a coin to obtain a bit β . He then sends the adversary $g_1^{d_0 s_1}$ and $g_1^{d_\beta s_2}$, where s_1, s_2 are secrets corresponding to keywords k_1, k_2 , as defined in our scheme. If $\beta = 0, d_\beta = d_0$ and the two values belong to the same document, whereas if $\beta = 1$, they belong to different ones.*

Learning phase: *The adversary can perform two types of queries. First, he can pick a set of keywords K' and request to get the encrypted index entries for a document that matches these keywords. I.e. the challenger has to produce $g_1^{d s_i}$ for each $i \in K'$. Second, he can pick a set of keywords K'' , such that $\exists k_i \in K$ with $k_i \neq k_1, k_2$ and obtain the decryption key $\{g_2^{a_{k_j}}\}_{k_j \in K''}$. The adversary can perform a polynomial number of either type of query.*

Response: *The adversary then outputs β' as his best guess for β .*

The two types of queries correspond to what Bob might be able to learn given access to the WORM drive. The first query corresponds to finding index entries corresponding to other documents with an arbitrary set of keywords. Such entries may be obtained by using per-query keys for K' , or through some external knowledge of what documents may have been stored during disposition period \mathcal{D} . The second query corresponds to obtaining keys to litigation holds that include some keyword other than k_1 or k_2 . Such keys cannot be used to decrypt the two challenge document IDs and to see if they match a real document.

We can define the advantage of an adversary playing the reconstruction game as:

$$P[\beta' = 0 | \beta = 0] - P[\beta' = 0 | \beta = 1]$$

Theorem 2 (Reconstruction Security). *For any polynomial-time adversary A that has advantage ϵ in the Reconstruction Game, there exists a polynomial-time adversary A' that has an advantage ϵ in solving the Decisional Diffie-Hellman problem in G_1 .*

Proof. We prove the theorem by simulation. Given an adversary A , we construct A' as follows.

A' takes as input $g_1, g_1^a, g_1^b, g_1^c \in G_1$, its goal is to determine whether $c = ab$.

A' starts by executing A to obtain the challenge keywords k_1, k_2 .

A' sets up the secret parameters by first picking random values r_i for $i \in \mathcal{K}$. It assigns $s_{k_1} = r_{k_1}, s_{k_2} = r_{k_2} b$, and $s_i = r_i(1 + b)$ for all others. It also picks a random r and sets $s = r(1 + b)$. Note that s_i are uniformly distributed for all i . It then supplies the public parameters to A .

$$\begin{aligned}
 g_1^{s_{k_1}} &= g_1^{r_{k_1}} \\
 g_1^{s_{k_2}} &= (g_1^b)^{r_{k_2}} \\
 g_1^{s_i} &= (g_1 \cdot g_1^b)^{r_i} \quad \text{for all other } i \\
 e(g_1, g_2)^s &= e(g_1 \cdot g_1^b, g_2^r)
 \end{aligned}$$

A' then challenges the adversary with $(g_1^a)^{k_1}$ and g_1^c to the adversary. Note that if $c = ab$, then $g_1^c = g_1^{as_2}$ and so $d_0 = d_\beta = a$. Otherwise, $d_\beta = c/b \neq a = d_0$.

For a query of type K' from A , A' picks a random document number d and then computes $g_1^{ds_j}$ for each $j \in K'$ as follows:

- If $j = k_1$, A returns $g_1^{ds_{k_1}}$, since both d and s_{k_1} are known to A
- If $j = k_2$, A returns $(g_1^b)^{dr_{k_2}} = g_1^{ds_{k_2}}$
- If $j \neq k_1, k_2$, A returns $(g_1 \cdot g_1^b)^{r_j d} = g_1^{dr_j(1+b)} = g_1^{ds_j}$

For a query of type K'' , A' needs to construct a set of a_j such that $\sum_{j \in K''} a_j s_j = s$. If $k_1, k_2 \notin K''$, then we need to simply find $\{a_j\}$ such that $\sum_{j \in K''} a_j r_j = r$. If $k_1 \in K''$ or $k_2 \in K''$, we can assign a_j 's by solving a system of two equations. As an example, consider $K'' = \{k_1, k_2, k_3\}$. In this case, we need:

$$a_1 r_{k_1} + a_2 r_{k_2} b + a_3 r_{k_3} (1 + b) = r(1 + b)$$

To find suitable a_j 's, we solve the following equations:

$$\begin{aligned}
 a_1 r_{k_1} + a_3 r_{k_3} &= r \\
 a_2 r_{k_2} + a_3 r_{k_3} &= r
 \end{aligned}$$

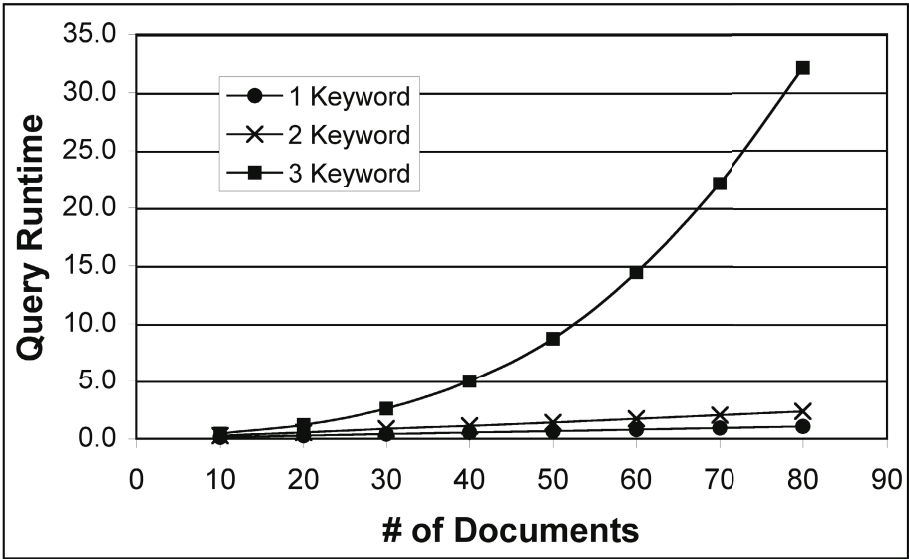
Since all the coefficients in the equations are known to A' , and there are three unknowns, A' can solve the equation and produce the correct key. In general, there will be at least two unknowns since in addition to k_1 or k_2 , K'' must contain at least one more $k_3 \neq k_1, k_2$, so A' can always solve the equation.

After A has made a polynomial number of queries and has output β , A' outputs that $c = ab$ if $\beta = 0$ and that $c \neq ab$ otherwise. Since A' will be correct whenever A is correct, they both enjoy the same advantage of ϵ .

6 Performance Evaluation

We carried out experiments both on micro-benchmark and on real-world data set to evaluate our scheme. The hardware platform used in our experiments was a 2.4 GHz 64-bit dual core Pentium IV machine running SUSE Linux. (Our tests were single-threaded, so only a single core was in use.)

In our micro-benchmark application, all the posting elements were of the same size and had the same set of document IDs. In other words, all the documents



(a) Query Runtime

Fig. 4. Runtime overhead of our scheme. x axis plots the number of documents. y axis shows the runtime in msecs.

satisfied the query. Figure 4 plots the total runtime as a function of the length of the posting list (number of documents). The different curves correspond to the different number of keywords in the query.

Based on these results, we have developed a model of the executing a query. As described earlier, a query involving q keywords requires the following:

- A bilinear map computation on each posting element of the q query posting lists. The total run time for this (b) is linear in the total length of the posting lists
- A product computation in the group G_T for each possible posting element combination of the q posting lists. The run time (p) for this is proportional to the number of such q posting element combinations, times the time for each such modulo computation (proportional to q). For example, for a 2-keyword query on two posting list of lengths l_1 and l_2 , the number of modulo computations is $l_1 * l_2$.

The total runtime (r) is a linear combination of the above two:

$$r = c_b * b + c_p * p$$

By applying least square fitting, we obtained the values of 12.34ms and 0.056ms for c_b and c_p respectively. The runtime predicted by the model was accurate within 3% in the average case and 6% in the worst case.

As a micro-benchmark, we used a collection of 422,000 emails from the Enron email corpus [8]. These emails were exchanged in the 2 year period from January 2000 to December 2001; we omitted the other 50,000 emails in the corpus because they were sprinkled very thinly across the time periods of 1994–2000 and 2002–3. Each email has a metadata tag identifying the sender, receiver, and the time the email was sent. We use this time information to divide the email documents into disposition groups of size 1 week. Noise terms are added to make the posting list same across the disposition groups to some threshold length. Assuming a uniform query model (each keyword equally likely to be queried) the average time for answering a 1, 2, 3 and 4 keyword query on each disposition group specific index was 0.8, 1.9, 18.2 and 1034s respectively. The worst case query execution times were 1.2, 3.0, 58.7s and 5461s respectively. This shows that queries with up to three keywords can be practically supported. The cryptographic operations are trivially parallelizable, so as multi-core systems become more common, the CPU time needed to execute queries should be significantly reduced.

Another important consideration is the size of the index. In our experiments, we had set the size of the group G_1 to 320 bits. Each posting element hence is $\frac{320}{8} = 40$ bytes long. Although this inverted index is substantially bigger than the unencrypted inverted index (it is about twice the size of the data corpus), this is not a serious consideration given the low cost of storage. The entire Enron email corpus is only a few GB in size, so the additional cost of the index storage is pennies under current prices.

Index creation requires one bilinear pairing computation for obtaining the document ID and as many modulo exponent computations as the number of keywords in the documents. On our platform, indexing a 100 keyword document takes about 200ms. Once again, multi-core architectures can be used to index documents in parallel.

7 Related Work

The first scheme for searches on encrypted data was proposed by Song et al. [19], with several improvements in following work [20,21,22]. Golle et al. extended the notion to that of a conjunctive query [23], with improvements in following work [24,25,26,27]. The conjunctive query work, while sharing similar goals to ours, uses a model where there are a fixed number of fields, each of which has a single keyword associated. The conjunctive query tests for the presence of a keyword in a particular position. Therefore, they work well for queries on structured fields (e.g., “From: Martha”, “To: Ralph”), but not full text indexing of records.

KP-ABE is one of several schemes proposed for attribute-based encryption [28,29,30,12,31,32]. Much of the work concerns creating efficiently expressing complex policies over attributes. As our construction is similar to KP-ABE [12], it is possible to extend it to support the more general access structures found therein, rather than simple conjunctive queries. However, since our approach requires the enumeration of index entry tuples that may satisfy the query, it would

negate the efficiency gains for complex policies. As discussed in Section 3.2, a simple application of attribute-based encryption is insufficient for our needs due to explicitly stated policies. Kapadia et al. have investigated hidden policies with ABE [33], but they resort to a semi-trusted third party to enforce this property.

8 Conclusion

We have developed a new type of encrypted index. Our index allows the owner to create a keys granting access to only a subset of the documents in the index that match a particular query. We showed how this scheme can be used to address the litigation hold problems of compliance storage, and also to limit potential abuse from auditors. We formally proved the security of our scheme and demonstrated by experiments that it is practical for large databases and moderately-sized queries. Our scheme may have applications beyond compliance storage to other shared databases where people are to be given restricted access based on attributes; it improves upon the Key-Policy Attribute-Based Encryption construction by being able to hide which attributes documents are encrypted under.

Acknowledgments

We would like to thank Ian Goldberg for his advice on the security proof and the anonymous reviewers for their helpful comments. This work was supported by an IBM Fellowship and NSF CNS 0716532.

References

1. The Enterprise Storage Group, Inc.: Compliance: The effect on information management and the storage industry (2003), <http://www.enterprisestoragegroup.com>
2. Securities and Exchange Commission: Guidance to broker-dealers on the use of electronic storage media under the national commerce act of 2000 with respect to rule 17a-4(f) (2001), <http://www.sec.gov/rules/interp/34-44238.htm>
3. Congress of the United States of America: Sarbanes-Oxley act (2002), <http://thomas.loc.gov>
4. EMC Corporation: EMC Centera content addressed storage system (2003), <http://www.emc.com/products/systems/centera.ce.jsp>
5. IBM Corporation: IBM TotalStorage DR550 (2006), <http://www-03.ibm.com/systems/storage/index.html>
6. Network Appliance, Inc.: SnaplockTM compliance and SnapLock enterprise software (2003), <http://www.netapp.com/products/filer/snaplock.html>
7. Mitra, S., Hsu, W.W., Winslett, M.: Trustworthy keyword search for regulatory-compliant records retention. In: Dayal, U., Whang, K.Y., Lomet, D., Alonso, G., Lohman, G., Kersten, M., Cha, S.K., Kim, Y.K. (eds.) Conference on Very Large Data Bases, VLDB Endowment, September 2006, pp. 1001–1015 (2006)

8. Cohen, W.W.: Enron email dataset (2005), <http://www.cs.cmu.edu/~enron>
9. Witten, I.H., Moffat, A., Bell, T.C.: *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco (1999)
10. Zhu, Q., Hsu, W.W.: Fossilized index: the linchpin of trustworthy non-alterable electronic records. In: *ACM SIGMOD International Conference on Management of Data*, pp. 395–406. ACM, New York (2005)
11. Mitra, S., Winslett, M., Borisov, N.: Deleting index entries from compliance storage. In: Kemper, A. (ed.) *Conference on Extending Database Technology* (March 2008)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Wright, R., di Vimercati, S.D.C. (eds.) *ACM Conference on Computer and Communications Security*, October 2006, pp. 89–98. ACM, New York (2006)
13. Frey, G., Rück, H.: A remark concerning m -divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation* 62(206), 865–874 (1994)
14. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory* 39(5), 1639–1646 (1993)
15. Miyaji, A., Nakabayashi, M., Takano, S.: New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 84(5), 1234–1243 (2001)
16. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. *Cryptology ePrint Archive*, Report 2005/417 (2005), <http://eprint.iacr.org/>
17. Joux, A.: A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology* 17(4) (2004)
18. Bellare, M., Canetti, R., Krawczyk, H.: Message authentication using hash functions: the HMAC construction. *CryptoBytes* 2(1), 12–15 (1996)
19. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: *IEEE Symposium on Security and Privacy*, pp. 44–55 (2000)
20. Goh, E.J.: Secure indexes. *Cryptology ePrint Archive*, Report 2003/216 (2003), <http://eprint.iacr.org/>
21. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, Springer, Heidelberg (2004)
22. Waters, B., Balfanz, D., Durfee, G., Smetters, D.: Building an encrypted and searchable audit log. In: *Network and Distributed System Security Symposium* (2004)
23. Golle, P., Staddon, J., Waters, B.: Secure Conjunctive Keyword Search over Encrypted Data. In: *International Conference on Applied Cryptography and Network Security* (June 2004)
24. Ballard, L., Kamara, S., Monrose, F.: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) *ICICS 2005*. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)
25. Park, D., Kim, K., Lee, P.: Public key encryption with conjunctive field keyword search. In: *WISA*, pp. 73–86 (2004)
26. Byun, J., Lee, D., Lim, J.: Efficient Conjunctive Keyword Search on Encrypted Data Storage System. In: Atzeni, A.S., Liyo, A. (eds.) *EuroPKI 2006*. LNCS, vol. 4043, pp. 184–196. Springer, Heidelberg (2006)

27. Hwang, Y., Lee, P.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) *Pairing 2007*. LNCS, vol. 4575, Springer, Heidelberg (2007)
28. Ostrovsky, R., Waters, B.: Attribute-based encryption with non-monotonic access structures. [34] 195–203
29. Chase, M.: Multi-authority attribute-based encryption. In: Vadhan, S.P. (ed.) *TCC 2007*. LNCS, vol. 4392, Springer, Heidelberg (2007)
30. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: *IEEE Symposium on Security and Privacy* (2007)
31. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R.J.F. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
32. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. [34], 456–465
33. Kapadia, A., Tsang, P., Smith, S.: Attribute-based publishing with hidden credentials and hidden policies. In: Arbaugh, W., Cowan, C. (eds.) *Network and Distributed Systems Security Symposium* (March 2007)
34. Syverson, P., Wright, R.: *The 14th ACM Conference on Computer and Communications Security*. ACM, New York (2007)

A Practical and Efficient Tree-List Structure for Public-Key Certificate Validation

Tong-Lee Lim¹, A. Lakshminarayanan¹, and Vira Saksen²

¹ Institute for Infocomm Research, A*STAR Singapore
{tllim,lux}@i2r.a-star.edu.sg

² ECE Department, National University of Singapore
u0508585@nus.edu.sg

Abstract. In this paper, we present the Tree-List Certificate Validation (TLCV) scheme, which uses a novel tree-list structure to provide efficient certificate validation. Under this scheme, users in a public-key infrastructure (PKI) are partitioned into clusters and a separate blacklist of revoked certificates is maintained for each cluster. The validation proof for each cluster's blacklist comes in the form of a hash path and a digital signature, similar to that used in a Certificate Revocation Tree (CRT) [1]. A simple algorithm to derive an optimal number of clusters that minimizes the TLCV response size was described. The benefits and shortcomings of TLCV were examined. Simulations were carried out to compare TLCV against a few other schemes and the performance metrics that were examined include computational overhead, network bandwidth, overall user delay and storage overhead. In general, we find that TLCV performs relatively well against the other schemes in most aspects.

1 Introduction

Certificate revocation and validation are important aspects of public-key infrastructures (PKIs). The digital certificate that binds a user's identity to a public-private keypair needs to be revoked when the private key has been compromised or due to other reasons which could cause a breach in security. Before using a public-key of another user to verify a digital signature, the digital certificate associated with the key must be validated to ensure that it has not been revoked. The same applies when one wishes to use a public-key of another user to encrypt a document. In order for certificate revocation and validation to achieve its purpose, the mechanism used must provide timely information to users and be scalable for deployment. In this paper, we introduce the Tree-List Certificate Validation (TLCV) scheme, which uses a hybrid tree-list structure that is efficient, practical and straightforward to implement.

One of the very first mechanisms proposed for certificate validation relied on the use of a blacklist that contains serial numbers of revoked certificates. Such a blacklist is referred to as the certificate revocation list (CRL) [2]. Most, if not all, internet browsers support the downloading of a CRL for certificate validation under the X.509 standard. One major drawback of using CRLs is that when the

number of revocations is large, a large amount of response data would have to be downloaded. In addition, the network bandwidth that the distribution server needs to support is also affected. To reduce the network bandwidth required for certificate validation, the Internet Engineering Taskforce (IETF) came up with the Online Certificate Status Protocol (OCSP) [3], in which a server known as an OCSP responder issues a signed response for each certificate validation request. While OCSP provides small responses and has low network bandwidth requirements, the need to compute a digital signature (which is an expensive operation) for every response makes it non-scalable when the number of requests is large. Variants of CRL have also been introduced for better performance. Examples include the segmented CRL, over-issued CRL, sliding window delta CRL [4], and more recently, the augmented CRL [5].

In [1], Kocher took a different approach by proposing a tree-based method known as the certificate revocation tree (CRT) that provides reasonably small responses for certificate validation. However, when changes to certificate status occur, the entire tree needs to be reconstructed, incurring a substantial amount of hash computations. In [6], Naor and Nissim introduced the authenticated dictionary (AD) that uses a 2-3 tree in place of a binary tree to provide greater efficiency when handling revocation updates. However, one shortcoming of AD is that it is complicated and not straightforward to implement.

Based on insights gained from these works, we propose the TLCV scheme, which uses a novel tree-list structure to provide better performance for certificate validation without the shortcomings faced by CRT or AD. We explain the proposed tree-list structure and describe a simple algorithm to derive an optimal tree height that minimizes the response size for TLCV. We also discuss the benefits and possible drawbacks when implementing TLCV, and compare its performance against other previously proposed schemes with the use of simulations. The simulations were carried out based on a constant-rate certificate arrival model, as well as a real-world model. Various aspects of performance, including computational overhead, network bandwidth, overall user delay and storage overhead, were examined. In general, we find that TLCV performs better than most of the other schemes in most aspects.

2 Background

The certificate revocation tree (CRT) introduced by Kocher is a certificate revocation and validation method that is based on a binary Merkle hash tree [1]. The CRT arranges the serial numbers of revoked certificates in an ordered manner and uses a path along the Merkle hash tree and a digitally signed root to verify the integrity and authenticity of the certificate validation response. This allows a certificate to be validated with a significantly smaller response as compared to CRL. However, one problem of this solution is that the entire CRT needs to be reconstructed when changes to certificate status occur, i.e. when there are new revocations that have to be inserted or when expired revocations need to be removed.

In [6], Naor and Nissim introduced an Authenticated Dictionary (AD) as an improvement over the CRT. The proposed AD is based on a 2-3 tree instead of a binary tree. Using a 2-3 tree allows updates to be performed without the need to reconstruct the entire tree. The insertion or deletion of a leaf node due to a change in certificate status will only involve a re-computation of the hash path for the associated leaf node. However, balancing a 2-3 tree is a complicated affair and putting the AD into implementation is not straightforward. In this paper, we introduce a practical and efficient tree-list structure that only requires re-computing hash paths for changed leaf nodes and does not involve complicated rebalancing of the tree. The number of hash paths that need to be recomputed is at most that required for an AD and the size of a hash path is guaranteed to be lesser than that of a CRT.

Other related works that have been carried out include works on the use of interval cover families [7], skip lists [8], the hash-based NOVOMODO [9], the QuasiModo Tree [10] and the Huffman Merkle Hash Tree [11]. Besides OCSP and the various CRL schemes, only NOVOMODO is known to be implemented and available for practical deployment in a PKI. In fact, we find that since the introduction of the CRT, there has been no significant improvement to tree-based certificate validation approaches without much implementation complications, such as that exhibited by AD.

In this paper, introduce the TLCV scheme, which uses a hybrid tree-list structure that is efficient, practical and straightforward to implement. We conduct experiments to compare the performance of TLCV against CRT, which TLCV is based on, and AD, which is an efficient improvement over CRT. In addition, we also compare TLCV against those schemes with known practical implementations – namely OCSP, ACRL [5] (the most efficient CRL variant known to date), and NOVOMODO. Here, we note that OCSP differs from the other schemes in that it is essentially an online mechanism that provides real-time certificate validation, while the rest are based on periodic updates to certificate status or revocation information. Nonetheless, we contend that it would be useful and interesting to investigate how TLCV fares against OCSP, especially since OCSP is an openly adopted certificate validation mechanism for the public internet.

3 The Tree-List Certificate Validation Scheme

3.1 Description of Scheme

The proposed Tree-List Certificate Validation (TLCV) scheme is based on partitioning users in a PKI into clusters. Under TLCV, the public-key certificate issued for each keypair would include a cluster number in addition to the serial number. The certificate authority (or revocation authority) maintains a separate blacklist of revoked certificates for each cluster. Similar to CRT and AD, a hash path, together with the cryptographically signed root of a hash tree, is used to provide integrity protection and origin authentication for each cluster's blacklist. The blacklist for each cluster is first hashed using a secure one-way hash function (e.g. SHA-1 or SHA-256) to produce the leaf node of a Merkle hash tree,

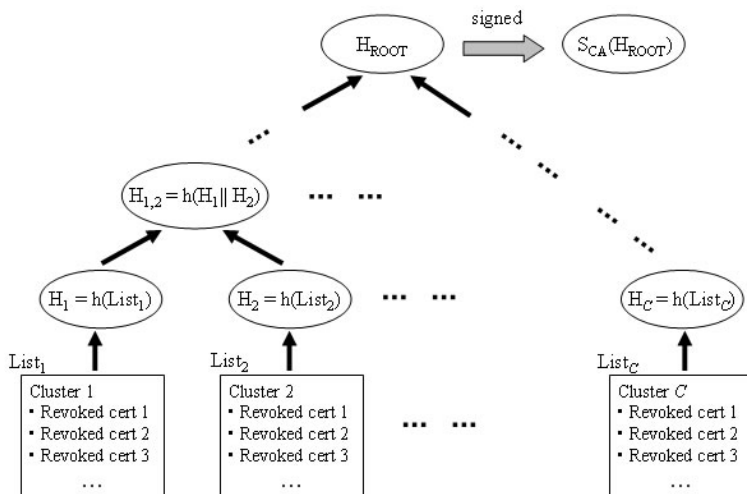


Fig. 1. The proposed tree-list structure under the TLCV scheme

<ul style="list-style-type: none"> • Header: IssuerName (32 bytes) Cluster Number (3) LastUpdate (6) NextUpdate (6)
<ul style="list-style-type: none"> • Data: Revoked Entries (10 bytes/entry) <ul style="list-style-type: none"> - Serial Number (3) - Revocation Date (6) - Reason Code (1)
<ul style="list-style-type: none"> • Proof: Siblings of Hash Path (20 bytes/node) <ul style="list-style-type: none"> - SHA-1 Hash (20) - Signed Root (RSA) (128)

Fig. 2. Format of a validation response under the TLCV scheme

as shown in Fig. 1. The rest of the tree is produced the same way as in a CRT, i.e. the value of each non-leaf node is computed by hashing a concatenation of the values of its children. The root of the Merkle hash tree is then signed (e.g. using RSA) together with a timestamp and other relevant information that is to be included in the validation response. The resulting digital signature and the hash path provided can then be used to verify whether the contents of the response have been received correctly and whether the response originated from the trusted certification authority.

With this scheme, each public-key certificate would now contain an additional field for the cluster number. During certificate validation, a user sends a request containing the cluster number of the certificate in question. The response

contains the blacklist for the cluster, the siblings of the hash path associated with the blacklist, and the signed root (see Fig. 2).

Under TLCV, the size of a response depends on the number of revoked certificates in a cluster and the size of the hash path. Let N_r denote the total number of revoked certificates in the PKI population and C denote the number of clusters (note that this is equal to the number of leaf nodes in the tree – in the case of a binary tree, $C = 2^h$ for some positive integer h). Assuming that the revoked certificates are uniformly distributed across the clusters, then the average number of revoked certificates in each cluster would be

$$N_c = \frac{N_r}{C} . \tag{1}$$

The size of each hash path in the tree is given by $h = \log_2 C$. Hence, the size of a response under the TLCV scheme effectively depends on C (or h) and N_r . In a study performed in [12], the authors showed that N_r reaches a constant value after a certain duration of time, as new revocations are balanced by revocations that expire. Furthermore, we note that in most PKI deployments, the PKI population is known and is more or less fixed. If the number of revocations at steady state is known, or if this can be predicted or estimated given the PKI population size, then we can use the information to determine a suitable value for C (or h) such that the size of the response is minimized. Henceforth, we shall assume that N_r refers to the number of revoked certificates at steady state.

3.2 Finding the Optimal Number of Clusters

Based on the response format shown in Fig. 2 and assuming that the revoked certificates are uniformly distributed across clusters, the size of a validation response under TLCV is

$$S_{resp} = S_{header} + S_{entry} \left(\frac{N_r}{C} \right) + S_{hash} (\log_2 C) + S_{sign} \tag{2}$$

When expressed in terms of the height of the tree, we have

$$S_{resp} = S_{header} + S_{entry} \left(\frac{N_r}{2^h} \right) + S_{hash} (h) + S_{sign} . \tag{3}$$

Our aim is to obtain the value of C or h that minimizes S_{resp} . Using (2), we solve for the value of C that satisfies

$$\min \left\{ S_{header} + S_{entry} \left(\frac{N_r}{C} \right) + S_{hash} (\log_2 C) + S_{sign} \right\} . \tag{4}$$

and obtain the optimal number of clusters and tree height as

$$C^* = \frac{S_{entry} N_r \log_e 2}{S_{hash}} . \tag{5}$$

$$h^* = \log_2 \left(\frac{S_{entry} N_r \log_e 2}{S_{hash}} \right) . \tag{6}$$

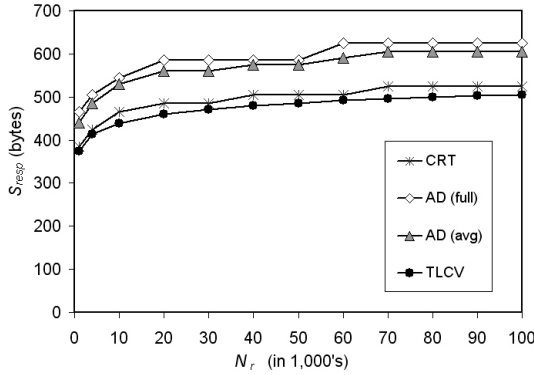


Fig. 3. Graph of response size S_{resp} against number of revoked certificates N_r for the various tree-based certificate validation schemes

However, the h^* derived from (6) may not necessarily be a positive integer. To ensure that this condition is met, we replace (5) and (6) with the following:

$$C^* = 2^{\lceil \log_2(S_{entry} N_r \log_e 2 / S_{hash}) \rceil} . \tag{7}$$

$$h^* = \left\lceil \log_2 \left(\frac{S_{entry} N_r \log_e 2}{S_{hash}} \right) \right\rceil . \tag{8}$$

In this case, (7) and (8) give approximations to the optimal number of clusters and tree height. We now describe a simple algorithm that allows us to obtain the precise optimal number of clusters that minimizes the response size:

1. Based on the known (or estimated) number of revoked certificates at steady state, compute h^* using (8).
2. Compute S_{resp} by substituting $h = h^*$ into (3) to obtain s . Repeat with $h = h^* - 1$ and $h = h^* + 1$ to obtain s^- and s^+ respectively.
3. (a) If $s \leq s^-$ and $s \leq s^+$, then h^* is the optimal tree height. Output $C = 2^{h^*}$.
 (b) Otherwise, obtain $s' = \min(s^-, s^+)$. The corresponding value of h , say h' , gives the optimal tree height. (This can be verified by checking that the values of S_{resp} obtained from $h' - 1$ and $h' + 1$ are greater than or equal to that obtained from h' .) Output $C = 2^{h'}$.

Using this algorithm, we can construct a tree-list structure that guarantees a low response size. Fig. 3 shows the graph of response size against N_r for TLCV compared with CRT and AD (in the average case and for a full ternary tree). The responses for CRT and AD are assumed to be of the format shown in Fig. 4. From the graph, we find that TLCV generally provides smaller responses than the other tree-based schemes.

3.3 Benefits and Drawbacks of TLCV

In this section, we discuss the benefits and possible drawbacks of TLCV. The benefits of TLCV can be listed as follows:

• Header:	IssuerName (32 bytes) LastUpdate (6) NextUpdate (6)
• Data:	2-tuple (S_1, S_2) (6) Revocation Date for S_2 (6) Reason Code (1)
• Proof:	Siblings of Hash Path (20 bytes/node) - SHA-1 Hash (20) Signed Root (RSA) (128)

Fig. 4. Format of a validation response under CRT and AD

- Just like CRT and AD, TLCV only requires the computation of a single digital signature during each update period. While additional hash operations need to be performed, the time taken for them is only a small fraction of that required for a signature. This is an advantage over on-demand schemes like OCSP, whereby a signature needs to be computed for every response, causing scalability problems when the number of requests get large.
- Changes to revocation entries only affect the path from the affected leaf node (corresponding to the cluster the revoked certificate is in) to the root node. In cases whereby multiple revoked certificates belong to the same cluster, only a single path needs to be updated for those revocations. Moreover, at the optimal number of clusters, the height of a TLCV is lower than that for a CRT or AD, i.e. the hash path for a TLCV is shorter than that for CRT and AD. Hence, during each update, the number of hash computations required for a TLCV is lesser than that required for a CRT or an AD.
- Unlike AD, there is no need to rebalance the tree for TLCV. The structure of the tree remains the same as only the contents of the leaf nodes change. The updating of a TLCV is straightforward and easier to implement than an AD based on a 2-3 tree.
- As witnessed in the previous section, the response size of TLCV is smaller than that of CRT or AD. Consequently, the network bandwidth required to support the download of validation responses under TLCV is expected to be smaller than that under CRT or AD.

On the other hand, some possible drawbacks to TLCV are as follows:

- There is a need to have an a priori knowledge of the PKI population size and revocation rate in order to determine the optimal number of clusters. While the PKI population size can be pre-determined in many cases, information on the revocation rate may not be readily available. Under such circumstances, an estimate for the revocation rate would have to be used.
- If the size of the PKI population changes drastically or the actual number of revoked certificates at steady state differs greatly from what was expected, the number of clusters may not be optimal anymore. This could possibly affect the performance of TLCV.

Table 1. Ranges of N_r and the corresponding h^* under optimal TLCV

N_r	h^*
1,024 - 2,047	9
2,048 - 4,095	10
4,096 - 8,191	11
8,192 - 16,383	12
16,384 - 32,767	13
32,768 - 65,535	14
65,536 - 131,071	15

- When deriving the optimal TLCV, it is assumed that the number of revoked certificates is uniform across different clusters. However, in a real-life scenario, clusters may have unequal numbers of revoked certificates. Thus, a client could end up downloading a larger response if the certificate to be validated belongs to a cluster that has a larger number of revoked certificates.

In general, the first two drawbacks would be more significant in a highly dynamic environment, where users join or leave the PKI in an unpredictable fashion and the number of users in the PKI varies drastically with time. In a fixed organization whereby the number of users is more or less stable, those drawbacks would have a lesser effect TLCV. With regards to the third drawback, we note that if the certificate to be validated is equally likely to be from any cluster, then the size of data downloaded will be averaged out due to clusters that have smaller number of revoked certificates (i.e. smaller responses). However, if the certificate to be validated is more likely to belong to a cluster containing a larger number of revoked certificates, the performance of TLCV would suffer.

3.4 Effects of Deviations to Expected N_r

In this section, we investigate how the TLCV scheme would be affected if the number of revoked certificates happens to differ from its expected value. We first investigate how changes in the total number of revoked certificates could affect the performance of TLCV by examining the optimal TLCV tree height h^* for various ranges of N_r . From Table 1, we see that if $h^* = h$ is the optimal tree height for $N_r = n$, then the optimal tree height for $N_r = 2n$ would be $h^* = h + 1$. In other words, if the actual value of N_r is double of what was expected, then the height of the TLCV tree would need to be increased by one in order for it to be optimal. (This is characteristic of a binary tree, where the number of leaf nodes in the tree doubles each time the height is increased by one). This suggests that on the average, TLCV would only deviate from optimality if the number of revoked certificates at steady state is double or more than double of what was initially expected.

Table 2 shows the response sizes for the various tree-based schemes, including the cases of TLCV where the tree height is not optimal (represented by TLCV#2,

Table 2. Comparison of response sizes (in bytes) for non-optimal TLCV

N_r	S_{resp}						
	TLCV	Non-optimal TLCV			CRT	AD	
		#2	#5	#10		(avg)	(full)
1,000	374	393	451	588	385	440	465
5,000	419	424	530	706	445	500	505
10,000	439	444	550	726	465	530	545
50,000	486	496	537	639	505	575	585
100,000	506	516	557	659	525	605	625

TLCV#5 and TLCV#10). TLCV#2 represents the case where the TLCV was designed to provide optimal response size for $\frac{1}{2}N_r$, i.e. the tree height is optimal for $\frac{1}{2}N_r$ (but not N_r). TLCV#5 and TLCV#10 represent the cases where the TLCV was designed to provide optimal response size for $\frac{1}{5}N_r$ and $\frac{1}{10}N_r$ respectively. The results under these columns correspond to the situation whereby the actual number of revoked certificate at steady-state is twice, five times and ten times (respectively) of what was initially expected for the PKI. From the table, we find that TLCV#2 performs reasonably well, with slightly larger response sizes than CRT but smaller than those achieved under AD. TLCV#5 has slightly larger response sizes than AD when N_r is below 50,000 but smaller response sizes for $N_r = 50,000$ and 100,000. TLCV#10 gives responses that are around 25-50% larger than the responses obtained under CRT and 5-30% larger than the responses obtained under AD. This shows that the performance of TLCV would only deteriorate to levels below that of AD if the actual number of revoked certificates at steady state is drastically larger (over five times more) than what was initially expected.

In summary, we find that TLCV can still perform at an optimal level if the number of revoked certificates deviate slightly from its expected value. Even when the deviation is large enough to cause the TLCV to become non-optimal, performance gains over the other schemes can still be achieved unless the deviation is drastically larger (over five times more) than the expected value.

3.5 Mitigating Drawbacks with Adaptation

To improve the performance of TLCV in a highly dynamic environment that causes N_r to deviate drastically from its expected value, adaptive changes to the tree-list structure can be made during the deployment of TLCV. The tree-list structure can be easily adapted to regain optimality in situations when the actual value for N_r is drastically different from its expected value. When N_r is overestimated, the resulting number of clusters would be greater than optimal. In order to compensate for this, we can combine the cluster lists to obtain a reduced tree, as shown in Fig. 5. The lists for two adjacent clusters are combined into a single new list and the leaf node (corresponding to the new list) of the resulting Merkle hash tree would be the parent of the original leaf nodes corresponding

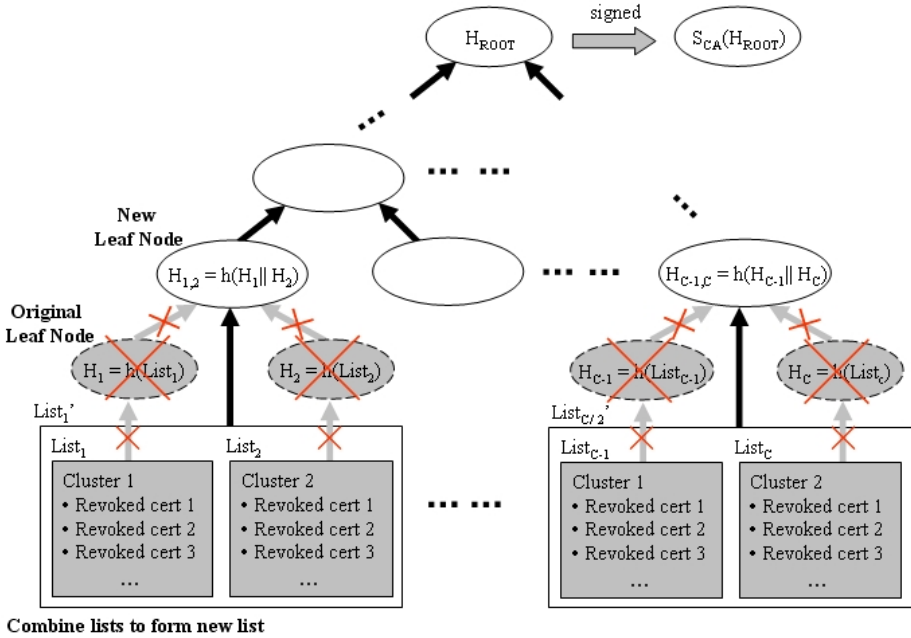


Fig. 5. Adapting the TLCV structure when N_r is overestimated

to the original lists. The new Merkle hash tree would simply be the original tree with its leaf nodes removed and the tree height reduced by one. Hence, the hash nodes are reused and there is no need to make any new hash computations. However, the new validation response would need to include an additional cluster number since both cluster numbers are required to identify the separate clusters within the new list. Based on Fig. 2, this would only incur an additional 3 bytes in the response.

On the other hand, if N_r is underestimated, there will be a need to increase the number of clusters and expand the tree, as shown in Fig. 6. In this case, the number of clusters is doubled and new leaf nodes are added for the revocation lists corresponding to the new clusters. New users joining the PKI would then be assigned to those new clusters when they apply for their digital certificates. The existing nodes from the original Merkle hash tree will be reused and need not be recomputed. The original root node now becomes the child of the new root and the height of the Merkle hash tree is increased by one. Hash values would have to be computed for the intermediate nodes that lie along the new paths between the new leaf nodes and the new root. We note that this adaptation is useful for PKIs where new membership can be expected and possibly result in an increase in the size of the PKI population. In PKIs whereby the users are fixed, then such an adaptation would not be applicable.

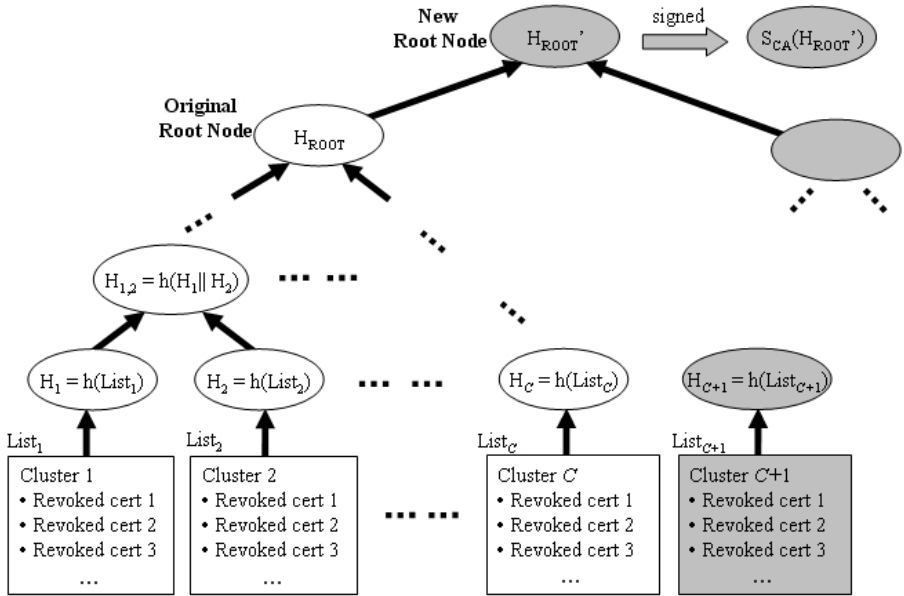


Fig. 6. Adapting the TLCV structure when N_r is underestimated

4 Performance Analysis

In this section, we conduct simulations in Matlab to analyze the performance of TLCV against a few other schemes, namely CRT, AD, OCSP, NOVOMODO and ACRL. We note that OCSP differs from the other schemes in that it is an online mechanism that provides real-time certificate validation, while the rest are based on periodic updates to certificate status or revocation information. While this is the case, we contend that it would be interesting to find out how TLCV fares against OCSP.

4.1 PKI Assumptions

We assume a PKI population of N users, whereby each user owns a single asymmetric key pair and a public-key certificate that binds the user to the key pair. Each certificate is valid for a duration L , after which it expires and is renewed (replaced by a new certificate) if it has not been revoked. Clients validate certificates (receive certificates that have not expired) at an average rate of v per client. Certificates are revoked at a constant rate r_r and revoked certificates expire at a rate r_x . In addition, the following assumptions were made:

- $L = 365$ days
- $v = 10$ certs/day
- $r_r = 0.05\% \times N = 0.0005N$ certs/day = r_x

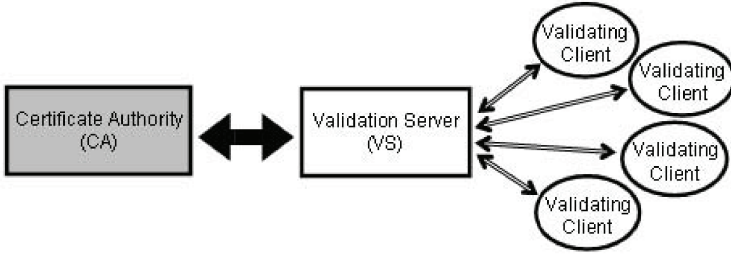


Fig. 7. The network architecture for certificate validation

As in [4], we assume further that the total number of certificates that have been revoked and have not yet expired at any one time is given by $N_r = \frac{1}{2}r_rL = 0.0913N$. If valid certificates expire at a rate r_v and the proportion of valid certificates that expire each day is the same as the proportion of revoked certificates that expire each day, then

$$\frac{r_v}{N} = \frac{r_r}{N_r}$$

$$\Leftrightarrow r_v = \frac{r_r}{N_r}N = 0.00548N \text{ certs/day}$$

The total number of unexpired certificates in circulation, is $N_{total} = N + N_r = 1.0913N$. Hence, when a client receives a certificate that has not expired, the probability that it is revoked is $P_r = \frac{N_r}{N_{total}} = 0.08$ and the probability that it is valid is $P_v = 1 - P_r = 0.92$.

4.2 Certificate Validation System Model

For the underlying certificate validation network, we assume a basic model shown in Fig. 7. The model comprises a CA and a validation server (VS) that clients connect to for certificate validation. All clients in the PKI are served by a single VS. When validating a certificate, the client sends a request to the VS and the VS replies with a response comprising the validation data, and a proof that provides evidence on the authenticity of the information. The response formats for the different schemes are assumed to be similar, except for the validation data and proof that is specific to each scheme. For TLCV, we assume the format shown in Fig. 2 and for CRT and AD, we assume the format shown in Fig. 4. The response formats for the other schemes are shown in Fig. 8.

For the arrival of digital certificates to clients, we consider two different models. The first model is the constant-rate model adopted by Cooper in [4]. In this model, each client receives certificates for validation at a constant rate v and the inter-arrival time between two successive certificates can be modelled using an exponential distribution with mean $\frac{1}{v}$. The second model is a real-world model that more accurately reflects the actual deployment and operation of an information technology (IT) infrastructure. It is based on the daily activity within

OCSP	
• Header:	IssuerName (32 bytes) EffectiveTime (6)
• Data:	Certificate Status (1) Revocation Date (6) Reason Code (1)
• Proof:	RSA Signature (128)

NOVOMODO	
• Header:	IssuerName (32 bytes) LastUpdate (6) NextUpdate(6)
• Data:	Certificate Status (1) Revocation Date (6) Reason Code (1)
• Proof:	SHA-1 Hash (20)

ACRL	
• Header:	Same as NOVOMODO (44 bytes)
• Data:	CRL Entries (10 bytes/entry) - Serial Number (3) - Revocation Date (6) - Reason Code (1) Expired Entries (3 bytes/entry) - Serial Number (3)
• Proof:	RSA Signature (128)

Fig. 8. Formats of validation responses for OCSP, NOVOMODO and ACRL

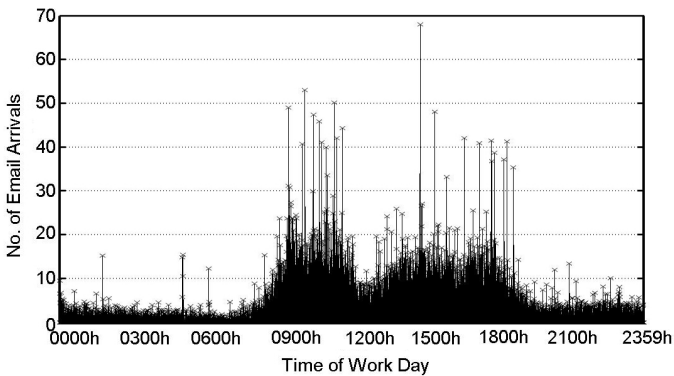


Fig. 9. The email arrival pattern (obtained by summing the email arrivals over 1 min intervals) for an organization of 700 employees on a single workday

an organization, i.e. the load during working hours is expected to be heavy while the load outside working hours would be light or close to zero. To model such a load pattern, we can either gather statistics on webpage requests made by users from a web server, or we can gather email arrival statistics from an email server. In this paper, we shall adopt the latter approach.

Statistics on email arrivals were collected from an email server in an organization of approximately 700 employees. The email arrivals for all work days in a single month were consolidated, from which the average email arrival pattern for a single work day was derived. Fig. 9 shows the resulting graph. This was extrapolated to model a PKI of N users, each validating certificates at an average rate v for each day.

4.3 Performance Metrics

In our performance analysis, we make use of some typical metrics that can be used to determine the scalability of each scheme for implementation and deployment in a PKI. The metrics are:

- **Peak computational overhead.** We assume that the main overhead comes from cryptographic computations and measure computational costs in terms of the total time taken for cryptographic operations. The following costs are assumed based on benchmarks obtained in [13]:
 - time to compute an RSA signature, $T_{sign} = 5$ ms
 - time to verify an RSA signature, $T_{verify} = 0.2$ ms
 - time to compute a SHA-1 hash, $T_{hash} = 0.3$ μ s
- **Peak network bandwidth required at VS.** Network bandwidth is obtained by computing the number of bytes transmitted from the VS to clients per unit time.
- **Peak delay experienced by a client.** This is measured by summing the peak computational time, the time taken to transfer the relevant data structures between the CA and the VS, the peak network delay between the VS and a client, and the computational time required at the client.
- **Storage overhead at VS.** This gives the amount of storage memory required at the VS.

4.4 Simulation Setup

Simulations were carried out on the optimized TLCV, as well as the non-optimal TLCV (labelled TLCV#), using Matlab. For TLCV#, we assume that the number of revoked certificates is under-estimated by one-fifth, i.e. the actual N_r is five times the estimated N_r . Other schemes that were also simulated include CRT, AD, ACRL, NOVOMODO and OCSP. For the periodic schemes (schemes other than OCSP), the interval for updating the revocation information at the VS was set to one minute in order to make certificate validation as close to real-time as possible without incurring an unreasonable overhead. Separate sets of simulations were conducted under Cooper's constant-rate model, as well as the

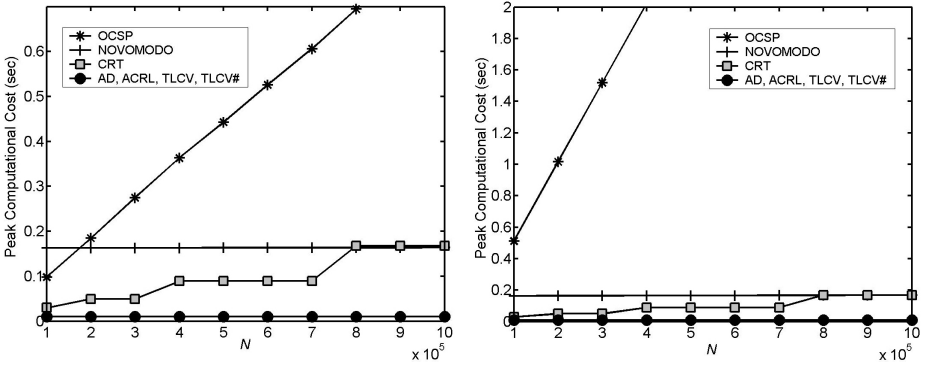


Fig. 10. Graph of peak computational overhead against N under the constant-rate model (left), and the real-world model (right)

real-world model, with the average rate of certificate arrival at each client set to $v = 10$ certs/day. The simulations were carried out for various values of N between 100,000 to 1 million.

4.5 Results and Discussion

Peak Computational Overhead. Fig. 10 shows results for peak computational overhead under Cooper’s constant-rate model and the real-world model. TLCV performs well since only one digital signature needs to be computed at every revocation update and the additional hash computations make up an almost negligible fraction of the time taken for computing a digital signature. The same goes for AD. ACRL has similar computational overheads since only one digital signature needs to be computed per update. CRT has higher computational costs than TLCV, TLCV# and AD due to the large number of hash operations required to reconstruct the entire tree. OCSP experiences the highest amount of computations due to the large number of digital signatures that have to be computed for the responses. NOVOMODO also experiences a relatively large computational overhead due to the computation of hash chains for new validity targets when expired certificates are renewed and revoked certificates are replaced.

Table 3. Number of hash operations per update for tree-based schemes

N	CRT	AD	TLCV	TLCV#
100,000	32,767	30	26	20
300,000	65,535	48	42	36
500,000	131,071	51	45	39
1,000,000	262,143	90	80	70

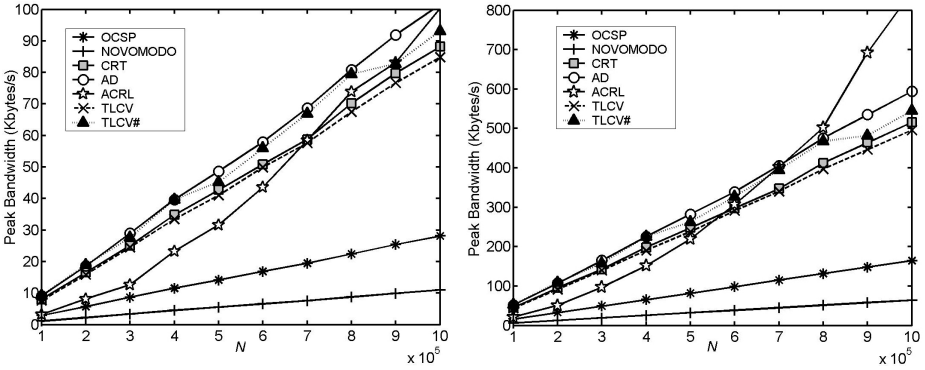


Fig. 11. Graph of peak network bandwidth required at the VS against N under the constant-rate model (left), and the real-world model (right)

Table 3 shows the number of hash operations required at the CA (similarly, at the VS) under the tree-based schemes during each update. From the table, we see that TLCV and TLCV# require lesser hash operations than both AD and CRT. This is because the height of a TLCV tree is generally smaller than that for CRT or AD, which results in a shorter hash path that needs to be re-computed when a change occurs. We note that the non-optimal TLCV# performs better than optimized TLCV because the optimization was carried out with respect to response size rather than computational overhead. In the following sections, we will see that response size has a greater effect on the performance of both schemes and TLCV has the edge over TLCV# in terms of overall performance.

Peak Network Bandwidth at VS. From Fig. 11, we find that TLCV requires lower network bandwidth than CRT and AD. This is due to its smaller response size, as witnessed in Table 2. While TLCV# requires slightly larger network bandwidth than CRT, it is still lesser than AD. Comparing TLCV against ACRL, we find that TLCV performs better for larger N (when $N \geq 700,000$ under Cooper’s constant-rate model and $N \geq 600,000$ under the real-world model). This is because for larger user populations, the size of the revocation list grows tremendously, causing the performance of ACRL to deteriorate. OCSP and NOVOMODO require smaller network bandwidths than TLCV due to their smaller response sizes, making them perform better than TLCV in this aspect.

Peak Delay at Client. Fig. 12 shows the peak delay experienced by a client. Under Cooper’s constant-rate model, we find that TLCV performs better than most of the schemes, losing out only to ACRL in the range of $100,000 \leq N \leq 600,000$. For the range $N \geq 700,000$, TLCV performs the best. Under the real-world model, TLCV performs better than than most of the schemes as well. In

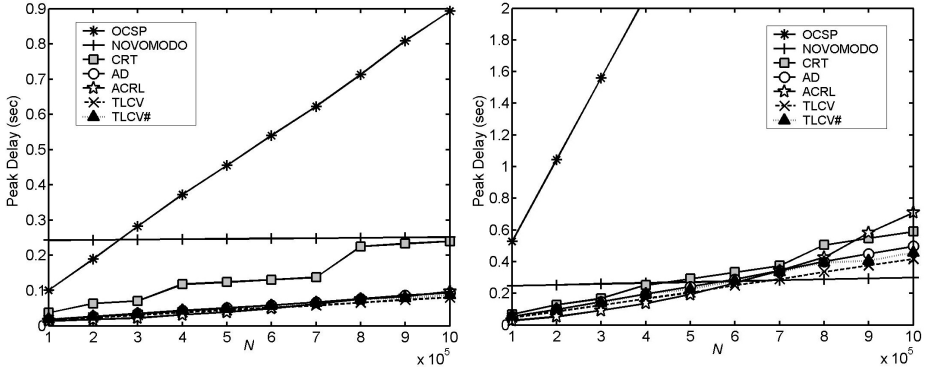


Fig. 12. Graph of peak delay against N under the constant-rate model (left), and the real-world model (right)

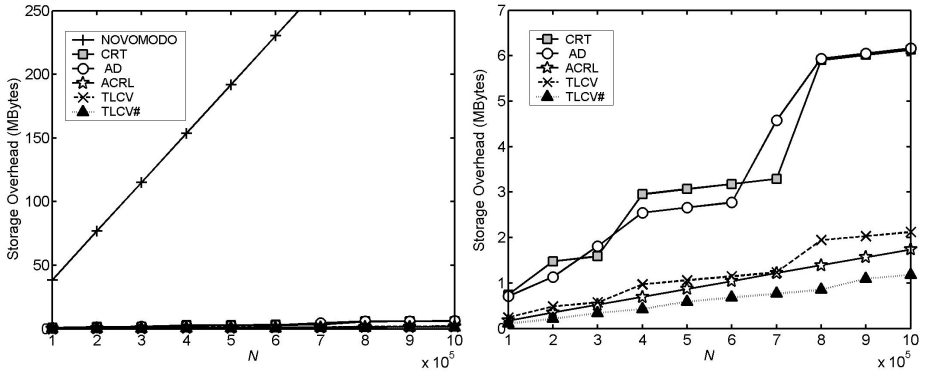


Fig. 13. Graph of storage overhead at VS against N . (The graph on the right is a scaled version of the graph on the left, with results for NOVOMODO excluded.)

the range $100,000 \leq N \leq 600,000$, only ACRL performs better and in the range $700,000 \leq N \leq 1$ million, only NOVOMODO performs better. These results show that in terms of overall performance (considering the sum of computational delay and network delay), TLCV performs better than most of the schemes. In fact, TLCV performs the best among the tree-based schemes.

Storage Overhead at VS. Fig. 13 shows results for storage overhead at the VS. OSCP has negligible storage overhead since it does not store any data at the VS. However, it requires a secure connection with the CA to retrieve revocation information from the CA and secure storage for the secret key that is used to sign the responses. This requires the VS to be trusted and secured against tampering, which implies extra overhead. On the other hand, TLCV, CRT, AD and ACRL can be deployed with untrusted VS since the data structures are

integrity-protected by the signature pre-computed by the CA. In general, we find that TLCV incurs lower storage overhead than CRT or AD and slightly larger storage overhead than ACRL. Non-optimal TLCV# incurs the lowest storage overhead among the various schemes but this comes at the expense of larger response size and overall delay.

Summary of Results. In general, we find that TLCV performs well compared against the other schemes. Among the tree-based schemes, TLCV requires lesser computations, network bandwidth, client delays and storage than CRT or AD. Comparing TLCV against OCSP and NOVOMODO, we find that while OCSP and NOVOMODO require smaller network bandwidths to support, the large computational overheads incurred cause them to perform poorly against TLCV. Moreover, both schemes require the VS to be trusted and secured against tampering since cryptographic secrets are kept at the VS. This incurs additional costs to these two schemes. Moreover, NOVOMODO requires an extremely large storage overhead. Comparing TLCV against ACRL, we find that both schemes perform closely in all four areas. In terms of network bandwidth and overall client delay, TLCV performs better for larger N and ACRL performs better for smaller N .

Considering results for non-optimal TLCV#, we find that while the network bandwidth and overall client delay deteriorates when TLCV deviates from optimality, this deterioration is not very significant even when the actual N_r is five times more than what was initially expected. TLCV# is still able to perform better than CRT, and performs close to AD. Moreover, TLCV# has a slight edge over the rest of the schemes in certain aspects, for example computational overhead and storage overhead.

5 Conclusion

We presented a novel tree-list structure for efficient certificate validation that partitions users into clusters and maintains a blacklist of revoked certificate for each cluster. We described a simple algorithm to obtain the optimal number of clusters that gives the minimal response size. We discussed the benefits and shortcomings of TLCV and performed simulations to compare its performance against a few other certificate validation schemes. From the results, we find that TLCV performs better than most of the other schemes in most aspects.

Acknowledgement

We would like to thank the numerous anonymous reviewers for their kind advice and valuable suggestions that have helped improve the quality of this paper.

References

1. Kocher, P.C.: On Certificate Revocation and Validation. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 172–177. Springer, Heidelberg (1998)
2. International Telecommunication Union, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework. ITU-T Recommendation X.509 (1197 E) (June 1997)
3. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public-Key Infrastructure Online Certificate Status Protocol - OCSP, RFC 2560 (June 1999)
4. Cooper, D.A.: A More Efficient Use of Delta CRLs. In: IEEE Symposium on Security and Privacy 2000, May 2000, pp. 190–202 (2000)
5. Lakshminarayanan, A., Lim, T.L.: Augmented Certificate Revocation Lists. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 87–98. Springer, Heidelberg (2006)
6. Naor, M., Nissim, K.: Certificate Revocation and Certificate Update. In: 7th USENIX Security Symposium (1998)
7. Blomer, J., May, A.: Key Revocation with Interval Cover Families. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 325–341. Springer, Heidelberg (2001)
8. Goodrich, M.T., Tamassia, R.: Efficient Authenticated Dictionaries with Skip Lists and Commutative Hashing, Technical Report, Johns Hopkins Information Security Institute (2000)
9. Micali, S.: NOVODOMO - Scalable Certificate Validation and Simplified PKI Management. In: Proceedings of the 1st Annual PKI Research Workshop (April 2002)
10. Elwailly, F.F., Gentry, C., Ramzan, Z.: QuasiModo: Efficient Certificate Validation and Revocation. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 375–388. Springer, Heidelberg (2004)
11. Munoz, J.L., et al.: Efficient Certificate Revocation System Implementation: Huffman Merkle Hash Tree (HuffMHT). In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2005. LNCS, vol. 3592, pp. 119–127. Springer, Heidelberg (2005)
12. Ma, C., Hu, N., Li, Y.: On the Release of CRLs in Public-Key Infrastructure. In: 15th USENIX Security Symposium, July 2006, pp. 17–28 (2006)
13. Speed Comparison of Popular Crypto Algorithms: Crypto++ 5.2.1 Benchmarks. URL: <http://www.eskimo.com/~weidai/benchmarks.html>

On the Security of the CCM Encryption Mode and of a Slight Variant

Pierre-Alain Fouque¹, Gwenaëlle Martinet², Frédéric Valette³,
and Sébastien Zimmer¹

¹ École normale supérieure, 45 rue d'Ulm, 75005 Paris, France

{Pierre-Alain.Fouque, Sebastien.Zimmer}@ens.fr

² DCSSI Crypto Lab, 51 Boulevard de la Tour-Maubourg F-75700 Paris 07 SP, France

Gwenaelle.Martinet@sgdn.pm.gouv.fr

³ CELAR, 35 Bruz, France

Frederic.Valette@dga.defense.gouv.fr

Abstract. In this paper, we present an analysis of the CCM mode of operations and of a slight variant. CCM is a simple and efficient encryption scheme which combines a CBC-MAC authentication scheme with the counter mode of encryption. It is used in several standards. Despite some criticisms (mainly this mode is *not online*, and requires *non-repeating nonces*), it has nice features that make it worth to study.

One important fact is that, while the privacy of CCM is provably guaranteed up to the birthday paradox, the authenticity of CCM seems to be guaranteed beyond that. There is a proof by Jonsson up to the birthday paradox bound, but going beyond it seems to be out of reach with current techniques. Nevertheless, by using pseudo-random functions and not permutations in the counter mode and an authentication key different from the privacy key, we prove security beyond the birthday paradox.

We also wonder if the main criticisms against CCM can be avoided: what is the security of the CCM mode when the nonces can be repeated, (and) when the length of the associated data or message length is missing to make CCM *on-line*. We show generic attacks against authenticity in these cases. The complexity of these attacks is under the birthday paradox bound. It shows that the lengths of the associated data and the message, as well as the nonces that do not repeat are important elements of the security of CCM and cannot be avoided without significantly decreasing the security.

Keywords: CCM, CBC-MAC, Counter mode.

1 Introduction

CCM stands for CTR + CBC-MAC and has been proposed by Doug Whiting, Russ Housley and Niels Ferguson. It is an *authenticated encryption* scheme based on the MAC-then-encrypt generic construction. It is interesting since it uses two very popular symmetric key schemes which are implemented in a lot of products and so, CCM can be constructed using “on the shelf” functions. It is used in

many standards of wireless networks such as IEEE 802.11 [22] (WiFi), IEEE 802.15.4 (Wireless Personal Area Network/ZigBee), standards of the internet in the RFC 3610 and RFC 4309 and finally in the NIST SP 800-38C [10].

The security of CCM is very interesting since it relies on some padding or formatting functions. Such requirements are not appreciated in general and cryptographers try to avoid such properties: for example, the security should not hold only because the length of the message is included in some message block. However, such specific requirements have been used before to construct hash function as in the Merkle-Damgard transformation of compression function to hash function or in order to make secure the CBC-MAC function for messages of arbitrarily length. It is a well-known property that messages that include their length in the first block are prefix-free and such property can be used to avoid classical attacks on the CBC-MAC.

CCM has also been criticized by some authors [19] who highlight three efficiency issues: “CCM is not on-line, CCM disrupts word-alignment, and CCM can’t preprocess static associated data”. The main issue is that CCM is not on-line since the sender has to know the length of the message before the beginning of the encryption. The two other critiques concern the associated data. Consequently, we have tried to see whether such criticisms can be avoided in the attack part of this paper.

1.1 Related Works

The security notions of symmetric encryption schemes have been intensively explored [2,3,5,9,14] and are now well understood. This background has allowed to design and analyze several operating modes [2,15,17,7] for symmetric authenticated encryption.

In this vein, two main authenticated encryption schemes with associated data were designed: AEX [7] and CCM [21]. They both are two-pass modes with non-repeating nonces and they both have been proved secure [12,7] until $2^{n/2}$ encryption queries for privacy *and* integrity. This bound is a classical bound and an encryption scheme secure up to this bound is commonly considered as secure. According to Jonsson, the privacy of CCM cannot be proved beyond the birthday paradox. However maybe the scheme is a good authentication scheme beyond this bound. At the end of his paper, Jonsson explains that if the CCM security is guaranteed until $2^{n/2}$ encryption queries, no attack which reaches this bound is known. Jonsson left as an open problem to fill the gap between the better known attack in 2^n encryption queries and this security bound. More precisely, he conjectures a security in 2^n encryption queries.

1.2 Our Results

The first part of our result concerns the presentation of an encryption scheme secure beyond the birthday paradox bound. We rely on CCM and propose a

slight variant of the CCM mode for which we are able to give a security proof beyond the birthday paradox for privacy and authenticity. We do not alter CCM too much to preserve some interesting properties. Precisely, we replace the block cipher used in the counter mode with a pseudo-random function. If one wants to base the security of the scheme on block cipher security, this pseudo-random function can be built using several block ciphers such as in [4,11,16]. Another alternative is to use the compression function of a hash function, where the key takes the place of the IV. This solution relies on the non classical assumption, that the compression function is a good pseudorandom function. However this assumption is more and more common [18] and is realistic. The privacy proof is a consequence of the privacy of the counter (CTR) mode when it is used with a random function. The authentication proof is built upon a method from [15] using the fact that with CTR, the encryption of the tag cannot be distinguished from a random bit string. Therefore one does not have to generate the tag to simulate the encryption.

In the second part of this paper, we try to justify why the non-repeating nonces and the length of the message and of the associated data are required for the security of CCM. All the attacks do apply to CCM and to the modified version that we propose, but we focus on the consequences for CCM, since CCM is standardized. We exhibit three attacks against the authenticity of the scheme. We, among others, worry about the “non-repeating” feature of the nonces. In a two party setting, it is easy to check such requirement since the two parties can maintain a counter. However, when several parties want to communicate to each other using the same key, it is difficult to maintain a global variable distributed among the participants. Consequently, the security of CCM with random nonces is an important issue.

In our first attack, we show a generic attack that requires $2^{(\ell+t)/2} + 2^\ell$ encryption messages, where ℓ is the nonces length and t is the length of the MAC. This attack is more theoretical than practical but it shows that the expected security bound of 2^n cannot be reached with random nonces.

Our second attack shows that when random nonces are used and when the length of the associated data is missing, $2^{\ell/2}$ encryption queries allows to forge a valid encrypted message (note that in practice $\ell < n$). It implies that if one want to remove the length of associated data to be able to preprocess static associated data, then one decreases CCM security under the proven birthday paradox bound.

Finally, our third attack shows that if random nonces are used and if the length of the message is not included in the padding function, then the authenticity of the scheme can be broken using $2^{2\ell/3}$ queries. It implies that if $\ell \leq 3n/4 = 96$ (which is realistic) and one wants to be able to make on-line encryption, then one decreases the security of CCM under the birthday paradox bound once more.

These attacks show that the security of CCM relies on the non-repeating nonce property and on the length of the message and of the associated data that is added before the message and make them prefix-free. This property is very useful to design secure encryption and authenticated schemes.

1.3 Organization

In section 2, we describe the CCM authenticated encrypted scheme. Then, we show our security proof beyond the birthday paradox for the authenticity and privacy in section 3. In Section 4, we describe some attacks that show why the non-classical assumptions, non-repeating nonces and prefix-free messages are important.

2 Security Notions

In the sequel, we briefly recall the basic security notions for blockciphers, pseudo-random functions, and symmetric encryption schemes. For the latter, we are interested into the integrity (we indistinctly use the words integrity and authentication in the sequel) and the privacy. The definitions we use are derived from [2,5].

Conventions. When an adversary A can interact with an oracle \mathcal{O} and at the end of the interaction outputs b , it is denoted by $A^{\mathcal{O}} \Rightarrow b$. If B and C are two events, the probability that the event B occurs, knowing the event C is denoted by $\Pr[B|C]$. When an adversary is involved in an event, the probability is considered upon the adversary random coins.

Let \mathcal{S} be a set of bit strings and let x and x' be a couple of bit strings from \mathcal{S} , we denote by $x \subset x'$ the fact that x is a prefix of x' . The set \mathcal{S} is prefix-free if for all couples $(x, x') \in \mathcal{S}^2$, $x \subset x'$ implies that $x = x'$. An adversary is said prefix-free if the set of the queries that it made to all the oracles, forms a prefix-free set. Finally, we denote by $\text{lsb}_k(x)$ the k least significant bits of x .

2.1 Pseudorandom Functions and Pseudorandom Permutations

Pseudorandom Permutations. Let $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a permutation family. We denote by \mathcal{S}_n the set of all the permutations from $\{0, 1\}^n$ to $\{0, 1\}^n$. The goal of a prp-adversary \mathcal{A} , which runs in time T , against E is to guess the value of b in the following game. The challenger chooses a bit b at random; if $b = 1$ he assigns π to a random permutation from \mathcal{S}_n otherwise he chooses a random key K in $\{0, 1\}^k$ and assigns π to $E(K, \cdot)$. The adversary can interact with π making up to q queries x_i and receives $\pi(x_i)$. The prp-advantage of \mathcal{A} , denoted $\text{adv}_E^{\text{prp}}(\mathcal{A})$, is:

$$\left| \Pr \left[\mathcal{A}^{E(K, \cdot)} \Rightarrow 1 \mid K \xleftarrow{\$} \{0, 1\}^k \right] - \Pr \left[\mathcal{A}^{\pi(\cdot)} \Rightarrow 1 \mid \pi \leftarrow \mathcal{S}_n \right] \right|.$$

Pseudorandom Functions. Let $F: \{0, 1\}^k \times \text{Dom} \rightarrow \{0, 1\}^t$ be a function family. We denote by Rand the set of all the functions from Dom to $\{0, 1\}^t$. The goal of a prf-adversary \mathcal{A} , which runs in time T , against F is to guess the value of b in the following game. The challenger chooses a bit b at random; if $b = 1$

he assigns f to a random function from Rand otherwise he chooses a random key K in $\{0, 1\}^k$ and assigns f to $F(K, \cdot)$. The adversary can interact with f making up to q queries x_i and receives $f(x_i)$. The prf-advantage of \mathcal{A} , denoted $\mathit{adv}_F^{\text{prf}}(\mathcal{A})$, is:

$$\left| \Pr \left[\mathcal{A}^{F(K, \cdot)} \Rightarrow 1 \mid K \xleftarrow{\$} \{0, 1\}^k \right] - \Pr \left[\mathcal{A}^f \Rightarrow 1 \mid f \leftarrow \mathcal{F}_{d,n} \right] \right|.$$

If \mathcal{A} is restricted to be prefix-free then its prf-advantage is called pf-prf-advantage and is denoted $\mathit{adv}_F^{\text{pf-prf}}(\mathcal{A})$.

2.2 Integrity

The security notion we use to define authenticity for a symmetric encryption scheme is the integrity of the ciphertext (denoted INT-CTXT). Formally, in the integrity game, the adversary \mathcal{A} is given access to an encryption oracle $\mathcal{E}(\cdot)$ and a verification oracle $\mathcal{VO}(\cdot)$ it can feed with queries of his choice. The encryption oracle encrypts the plaintext and answers by the corresponding ciphertext. The adversary feeds the verification oracle with a ciphertext, also called forgery attempt in the sequel, and the oracle answers 1 if the ciphertext is valid and 0 otherwise. The adversary goal is to generate a valid ciphertext (that is accepted by the verification oracle) which is different from all ciphertexts previously generated by the encryption oracle. Note that the adversary can send several queries to the verification oracle. The success probability of \mathcal{A} is:

$$\mathit{Succ}_{CCM}^{\text{int-ctxt}}(\mathcal{A}) = \Pr[\mathcal{VO}(C) \Rightarrow 1 \mid \mathcal{A}^{\mathcal{E}(\cdot), \mathcal{VO}(\cdot)} \Rightarrow C].$$

2.3 Privacy

The security notion used to define privacy for a symmetric encryption scheme is the indistinguishability security under chosen plaintext attacks (denoted IND-CPA). Formally, in the privacy game an adversary \mathcal{A} is given access to an encryption oracle $\mathcal{E}(\cdot)$ it can feed with queries of the form (M_0, M_1) where M_0 and M_1 are messages of his choice. At the beginning of the game this oracle chooses a bit b and always encrypts the message M_b . The adversary’s goal is to guess b , that is to say to distinguish the two cases. The indistinguishability is defined in the “left or right model” which has been introduced and proved to be the strongest one in [2]. The advantage of \mathcal{A} is:

$$\mathit{adv}_{CCM}^{\text{ind-cpa}}(\mathcal{A}) = \left| \Pr[\mathcal{A}^{\mathcal{E}(\cdot)} \Rightarrow 1 \mid b = 1] - \Pr[\mathcal{A}^{\mathcal{E}(\cdot)} \Rightarrow 1 \mid b = 0] \right|.$$

3 CCM Description

In this part, we describe the original CCM authenticated encryption mode and the format of its various inputs. In [21] recommendations are also given on various choices that have to be made to implement CCM: unique key for both the CBC chain and the counter mode, nonces that cannot be repeated. . . Some of these restrictions can be ignored without any security problems although some other are needed for security reasons. At the end of this part we discuss these choices.

3.1 Notations

In this paper, the following notations will be used:

- for any string or integer x , $|x|_2$ denotes its bit length, $|x|_8 = \left\lceil \frac{|x|_2}{8} \right\rceil$ its octet length, and $[x]_s$ denotes the binary representation of x on s bits;
- E is a block cipher with n -bit blocks and k -bit keys, where $n = 128$;
- M is a plaintext, consisting of blocks of n bits denoted M_1, \dots, M_{m-1} and a last block M_m with at most n bits.
- the associated data (data which is authenticated and not encrypted) is denoted by D_1, \dots, D_a and consists in $a - 1$ blocks of n bits and one block of at most n bits;
- the ciphertext C consists in $m + 1$ blocks C_0, C_1, \dots, C_m where C_i is n -bit long for $0 \leq i \leq m - 1$ and C_m is at most n bits;
- $B = B_0, B_1, \dots, B_r$ is the n -bit long formatted input used for the CBC-MAC computation, B_0 is called the pre-initial value;
- A_0, A_1, \dots, A_m are the inputs for the counter mode;
- the nonce used to derive the pre-initial value B_0 and the counter values A_0, A_1, \dots, A_m is denoted by N . This nonce is ℓ -bit long, with $7 \leq \ell/8 \leq 13$ (ℓ has to be divisible by 8);
- q is an integer such that $2 \leq q \leq 8$ and $q + \ell/8 = 15$, let Q denotes the bit representation of the octet length of the message M over q octets, *i.e.* $Q = \left[[M]_8 \right]_{8q}$;
- t denotes the bit length of the MAC, it has to be divisible by 16, and $4 \leq t/8 \leq 16$.

3.2 CCM Mode

The CCM mode can be basically viewed as an authenticate-then-encrypt composition instantiated with a CBC-MAC and a counter mode. The mode uses a block cipher E both in the CBC chain and in the counter mode. The block length is equal to $n = 128$ bits. We denote by K the key used in the CBC-MAC and by K' the one used in the counter mode. The choice of K and K' is discussed in section [3.4](#).

Let $M = M_1 \| M_2 \| \dots \| M_m$ be a plaintext and $D = D_1 \| D_2 \| \dots \| D_a$ associated data that will only be authenticated and not encrypted.

At first, the encryption box chooses a nonce N of ℓ bits. This nonce will be used to derive both the pre-initial value for the CBC-MAC and the counter blocks.

In a first step, the associated data and the plaintext blocks are authenticated. This consists in computing their CBC-MAC value. This computation is however quite different from the classical one: it authenticates a formatted input $B = B_0 \| \dots \| B_r$ derived from N , M , and D . The format of B is described in section [3.3](#). The 1-block pre-initial value B_0 is treated as the first message block in the CBC chain. Its main property is that it contains the tag length, the

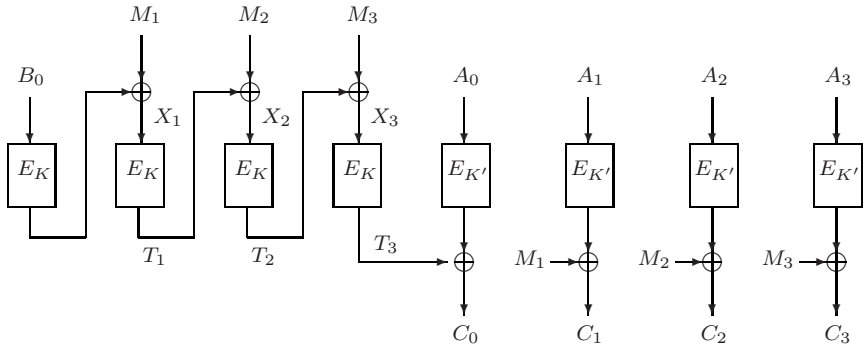


Fig. 1. CCM authenticated encryption mode

plaintext length and the nonce value. The CBC-MAC is a simple CBC chain without retail. Its output is denoted T and is t -bit long, with $32 \leq t \leq n$.

In a second step, the MAC value T and the plaintext M are concatenated and encrypted with a counter mode. The inputs for the counter mode are blocks of n bits denoted $A_0 \| A_1 \| \dots \| A_m$. Their format is described in section 3.3. Briefly, each one contains some flag information, the nonce value, and the index of the plaintext block. The tag T is encrypted as $C_0 = \text{lsb}_t(E_{K'}(A_0)) \oplus T$ and for all plaintext blocks $M_i, 1 \leq i \leq m, C_i = E_{K'}(A_i) \oplus M_i$.

The ciphertext $C = C_0 \| C_1 \| \dots \| C_m$ and the associated data D are then transmitted. The nonce value is transmitted if necessary (in case of non synchronizing).

Figure 1 describes the CCM mode for 3-block plaintexts and no associated data.

The decryption process consists in decrypting C with the counter mode and then to compute the tag T' with the formatted input B computed from the nonce N , the associated data and the recovered plaintext. If valid, the plaintext is returned. Otherwise an error message is given. For a description of CCM encryption and decryption algorithms in pseudo-code, see appendix A.

3.3 Formatting the Inputs

The specification [21] precisely describes the format for the different inputs.

The CBC-MAC chain uses a formatted input $B = B_0, \dots, B_r$. The n -bit pre-initial value B_0 is determined by a nonce of ℓ bits, and various information. The first octet is a flag one containing one bit for future extension, one bit to indicate whether associated data are present or not, three bits to indicate the octet length of the MAC value (which is necessarily different from 000) and three bits for the octet length of the binary representation of the octet length of the plaintext M . The remaining octets contain the nonce value followed by the value q , the bit

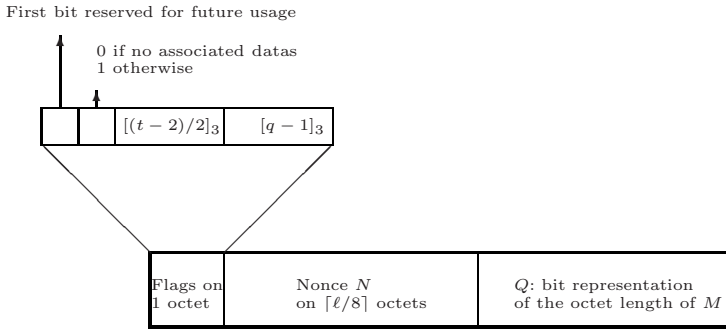


Fig. 2. The format of the pre-initial value B_0 for the CBC-MAC

representation of the octet length of M . Picture 2 represents the format of B_0 . Note that a collision on the B_0 values occurs if and only if the nonces collide, associated data are either used or not for both and the plaintexts are of the same octet length.

If there are authenticated data then let $B_1 \parallel \dots \parallel B_u$ be the concatenation of a particular encoding of the D size (for more details see [21]), of D , and of as few '0' as possible such that the resulting bit string can be partitioned into n -bit blocks (if there is no authenticated data $B_1 \parallel \dots \parallel B_u$ is the empty bit string). Let $B_{u+1} \parallel \dots \parallel B_r$ be the concatenation of M and of as few '0' as possible such that the resulting bit string can be partitioned into n -bit blocks. Remark that the encoding of B is made in such a way that the set of all possible formatted inputs B is prefix-free !

Finally, the inputs for the counter mode A_0, A_1, \dots, A_m are encoded as follows: the first octet contains some flag information (2 bits reserved for a future usage, three bits fixed to 000 and three bits containing the binary representation of $q - 1$). The remaining ones contain the nonce value, already used for formatting B_0 and the block number.

3.4 NIST Requirements

The CCM specification [21] gives indications to guide implementation choices. Of course, CCM should be used with the AES so the key length is either 128, 192, or 256 bits. The block length is 128 bits.

The CCM specification also provides a requirement for the key choice. Indeed, the same key should be used for both the CBC-MAC and the counter mode. Such a choice is of course debatable since it goes against the common sense based on the key usage separation. However, the security proof given by Jonsson in [12] is made for this case and ensures that CCM provides privacy and authenticity up to $2^{n/2}$ block cipher calls. Thus, choosing the same key for the two modes combined in CCM is not a security concern since the security bound is very close to the one given for a lot of other authenticated encryption modes [18,13].

However, even if this requirement is quite understandable in case of same security results, it becomes cumbersome if the security with two keys is much better than for a single key. In our modified CCM we can achieve a much better security with two keys. That is why in section 4, we focus on the security results for our modified CCM in case of non repeating nonces with two different keys.

Another requirement made in the CCM specification concerns the choice of the nonce values. It is explicitly defined that “*the counter blocks must be distinct within a single invocation and across all other invocations of the CTR mode under any given key*”. This is done by imposing non-repeating nonces. This requirement is here largely understandable: indeed, a collision on the nonce values can often be exploited to forge a valid ciphertext or to attack the privacy of the scheme. However, in practice, non repeating nonces could be very difficult to handle, particularly in a scenario where three or more users share the same key used with CCM. Thus, it can be interesting to carefully look at the CCM security when nonces are randomly chosen and can thus collide. This is done in section 5.

4 Modified CCM

4.1 Description

In this section we propose a modified version of CCM (mCCM) which we prove secure beyond the birthday paradox bound. The main difference between the original and the modified CCM versions is the use of a pseudorandom function to encrypt the tag and the message. Let F be a pseudorandom function family from $\{0, 1\}^n$ to $\{0, 1\}^n$, E a blockcipher over $\{0, 1\}^n$ and let K and K' be two keys chosen independently.

To encrypt a message M, D , a nonce is chosen, the corresponding formatted input is deduced and a CBC-MAC tag T is computed with the blockcipher E_K . Then, the ciphertext is computed as $C_0 = T \oplus \text{lsb}_t(F_{K'}(A_0))$ and $C_i = M_i \oplus F_{K'}(A_i)$.

The decryption process consists in decrypting C with the counter mode and then to compute the tag T' with the formatted input B' computed from the nonce N , the associated data and the recovered plaintext. If valid, the plaintext is returned. Otherwise an error message is given.

We remind that in this modified version, as in the original version, we impose non repeating nonces. This implies that adversaries against modified CCM can choose the nonce to encrypt a query, as soon as, any new nonce is different from all the previous one. However, for the verification queries the adversary is allowed to use a nonce which was already used in a previous encryption query.

In the following we prove the IND-CPA and INT-CTXT security of this modified version of CCM. Note that as proven in 5 these two security notions imply the IND-CCA security (with a tight reduction), which means that this protocol achieves the best security levels for privacy and for integrity (see 5 for precise definitions and relations between these notions).

4.2 Privacy

The modified CCM privacy is a direct consequence of the privacy of CTR using a pseudorandom function. This security result has been stated in [2]:

Theorem 1 (BDJR). *Suppose F is a PRF family from $\{0, 1\}^n$ to $\{0, 1\}^n$. Then, for any adversary A against privacy of CTR mode, with running-time T , and which can do at most q_e encryption queries of at most s blocks, there exists a prf-adversary A' against F with running time T and which can make at most sq_e queries, such that:*

$$\text{adv}_{CTR}^{\text{ind-cpa}}(\mathcal{A}) \leq \text{adv}_F^{\text{prf}}(\mathcal{A}').$$

The security of mCCM is an easy consequence of this theorem:

Theorem 2. *Suppose F is a PRF family from $\{0, 1\}^n$ to $\{0, 1\}^n$. Then, for any adversary \mathcal{A} against privacy of modified CCM mode, with running-time T , and which can do at most q_e encryption queries of at most s blocks, there exists a prf-adversary \mathcal{A}' against F with running time T and which can make at most $(s + 1)q_e$ queries, such that:*

$$\text{adv}_{mCCM}^{\text{ind-cpa}}(\mathcal{A}) \leq \text{adv}_F^{\text{prf}}(\mathcal{A}').$$

4.3 Integrity

To prove the integrity of ciphertexts (INT-CTXT) in modified CCM, we need the two following results. The first one is shown in [6] and upper bounds the advantage of a pf-prf adversary against CBC-MAC.

Theorem 3 (BPR). *Let \mathcal{A} be a prefix-free prf-adversary against the n -bit block CBC-MAC, \mathcal{A} can make at most $q \geq 2$ queries of at most s blocks and has a running-time of at most T . Then we have:*

$$\text{adv}_{CBC-MAC}^{\text{pf-prf}}(\mathcal{A}) \leq \frac{sq^2}{2^n} \left(12 + \frac{64s^3}{2^n} \right).$$

The second result comes from [3] and shows that if a protocol is INT-CTXT secure against an adversary which can make at most one verification query, then it is secure against adversaries which can make several verification queries.

Lemma 1. *Let \mathcal{A} be an adversary against the authenticity of a symmetric encryption schemes Π . Assume that \mathcal{A} makes at most q_e encryption queries and q_v verification queries all of at most s blocks. Then there exists an adversary \mathcal{A}' against the authenticity of Π , such that \mathcal{A}' makes at most q_e encryption queries and 1 verification queries, all of at most s blocks and:*

$$\text{Succ}_{\Pi}^{\text{int-ctxt}}(\mathcal{A}) \leq q_v \cdot \text{Succ}_{\Pi}^{\text{int-ctxt}}(\mathcal{A}').$$

The adversaries \mathcal{A} and \mathcal{A}' have the same running-time.

Thanks to these results, we can upper bound the advantage of any adversary against the INT-CTXT of mCCM.

Theorem 4. *Let \mathcal{A} an adversary against the authentication of mCCM, with running-time at most T , which can make at most q_e encryption queries of at most s blocks and q_v verification queries of at most s blocks. Then its success probability is upper bounded by:*

$$\text{Succ}_{mCCM}^{\text{int-ctxt}}(\mathcal{A}) \leq q_v \left(\frac{48(s+1)}{2^n} + 256 \left(\frac{(s+1)^2}{2^n} \right)^2 + \frac{1}{2^t} \right) + \text{adv}_F^{\text{prf}}(\mathcal{A}_1) + \text{adv}_E^{\text{prf}}(\mathcal{A}_2),$$

where \mathcal{A}_1 is a prf-adversary against F with running-time at most T , which can make at most $(s+1)(q_e + q_v)$ queries and \mathcal{A}_2 is a prp-adversary against E with running-time at most T , which can make at most $(s+1)(q_e + q_v)$ queries.

Proof. The following proof is a game-based proof. In the first game (game 1) the adversary faces the verification and encryption oracles which are simulated respecting strictly the protocol. In each new game we alter a bit the way we simulate the two oracles, so that in the last game we are able to upper bound the adversary success probability. Since we alter the simulation between two games, the adversary success probability is modified, so we have to upper bound this modifications; this upper bound is called the distance between two games. See [20] for details.

In the games 2 and 3 we replace successively the PRF F and the PRP E with respectively a true random function and a true random permutation. One can easily shows that there exists two adversaries \mathcal{A}_1 and \mathcal{A}_2 as stated in the theorem such that the distances between the games can be upper bounded respectively by $\text{adv}_F^{\text{prf}}(\mathcal{A}_1)$ and $\text{adv}_E^{\text{prf}}(\mathcal{A}_2)$.

Let \mathcal{A}_3 be the adversary against the authenticity of mCCM in the game 3, it can make q_e encryption queries and q_v verification queries, all of at most s blocks. Let \mathcal{A}' be an adversary against the authenticity of mCCM which makes q_e encryption queries of at most s blocks and 1 verification query of at most s blocks such that $\text{Succ}_{mCCM}^{\text{int-ctxt}}(\mathcal{A}_3) \leq q_v \cdot \text{Succ}_{mCCM}^{\text{int-ctxt}}(\mathcal{A}')$ (it exists thanks to lemma [1]). To upper bound the success probability of \mathcal{A}_3 , we upper bound the success probability of \mathcal{A}' . For this, thanks to \mathcal{A}' we construct \mathcal{D} a prefix-free prf-adversary against CBC-MAC with 2 MAC queries of at most s blocks, and then relate the success probability of \mathcal{D} with the one of \mathcal{A}' .

As described in the prf security definition, the prf-distinguisher \mathcal{D} faces a CBC-MAC oracle. To construct \mathcal{D} , we run \mathcal{A}' and to every of its encryption query (N^i, M^i, D^i) , we answer:

- \perp if there exists $k < i$ such that $N^i = N^k$,
- $(N^i, D^i, C^i = C_0^i || \dots || C_{m_i}^i)$ with $C_0^i \xleftarrow{\$} \{0, 1\}^n$, $C_k^i = M_k^i \oplus F(A_k^i)$.

To \mathcal{A}' verification query $(N, D, C = C_0 || \dots || C_m)$, we answer:

- if $N \neq N^k$ for all $k \leq q_e$, then we choose randomly a t -bit string T and compute the m -block message M with $M_i = C_i \oplus F(A_i)$; we give B , where B is the corresponding formatted input, to the CBC-MAC verification oracle which answers with T' and we reply to \mathcal{A}' with 1 if and only if $T = \text{lsb}_t(T')$,
- if there is $k \leq q_e$ such that $N = N^k$, $D = D^k$, and $C = C^k$, then we answer \perp ,
- if there is $k \leq q_e$ such that $N = N^k$, but $D \neq D^k$ or $C \neq C^k$, then, we compute the message blocks $M_i = F(A_i) \oplus C_i$, deduce the corresponding formatted input B ; we send the k^{th} formatted input B^k (from the k^{th} encryption query) to the CBC-MAC oracle and receives the corresponding tag T^k ; then we compute the tag of B : $T = \text{lsb}_t(T^k) \oplus C_0 \oplus C_0^k$; finally we send B to the CBC-MAC oracle, receives back T' , check if $T = \text{lsb}_t(T')$ and forward the answer to \mathcal{A}' (Note that since $D \neq D^k$ or $C \neq C^k$, $B \neq B^k$ and since the formatted inputs form a prefix-free set, the two queries made to the CBC-MAC oracle are prefix-free).

At the end, \mathcal{D} decides that it faces a true CBC-MAC oracle if the answer to the \mathcal{A}' verification query is equal to 1.

As soon as the nonces are different from each other, for a mCCM attacker the C_0^i are randomly distributed, therefore the answers to \mathcal{A}' are well simulated. Since there is no collision between the nonces, the probability of success of \mathcal{A}' is exactly the probability that \mathcal{D} outputs 1 when it faces a true CBC-MAC oracle. When \mathcal{D} faces a random function, its success probability is $1/2^t$, therefore we have:

$$\text{Succ}_{m\text{CCM}}^{\text{int-ctxt}}(\mathcal{A}') \leq \frac{1}{2^t} + \text{adv}_{\text{CBC-MAC}}^{\text{pf-prf}}(\mathcal{D}).$$

We remind that pf in pf-prf stands for prefix-free. Theorem 3 allows us to conclude. \square

In practice, we can consider that $s \leq 2^{40} - 1$ (in fact this is probably still a large upper bound of s). In the following, we omit the PRF and PRP advantages for simplicity reasons, since these terms are identical in the two bounds. Let assume that $t \geq 82$, previous theorem gives an upper bound of the integrity adversaries of $q_v \cdot 2^{-80}$. For the same values, Jonsson theorem [12] gives a security of approximately $(q_v + q_e)^2 \cdot 2^{-48}$. Remark that for a value of $t = 82$, our bound is tight since the simple attack which consists in trying to guess the CBC-MAC value has a success probability of $q_v/2^t = q_v \cdot 2^{-82}$.

5 Random Nonces

In this part we consider that the nonces used for the CCM mode are chosen at random in $\{0, 1\}^\ell$. In this case, collision between two random values are possible and such an event can be exploited by an adversary to forge a valid ciphertext. Of course, confidentiality cannot be ensured as soon as two nonces collide. Indeed, if such a collision occurs, the adversary can easily distinguish which plaintext is encrypted and then break the scheme in the sense of semantic security. However, forging a valid ciphertext, *i.e.* breaking the ciphertext unforgeability, is a different task. Attacking the privacy is independent and does not imply any weakness on

the integrity, even if, when one is compromised, the other often too. However, the technique used here to forge a valid ciphertext is completely different from the one used to break the semantic security in case of collision between nonces.

Note that the following attacks do apply to both original CCM and modified CCM, as long as the nonces are random. We focus our analyze to CCM because it is a standard, but the same conclusions could be stated for the modified version of CCM. Besides, remark that the security model is slightly different from previous section. As the nonce may repeat, the adversary can make as many queries as it wants, whereas in previous section the adversary was restricted to 2^ℓ encryption queries, the number of possible nonces. However, when in previous context the nonces used for encryption queries can be chosen by the adversary, as long as there are all distinct, in this context the nonces are chosen randomly by the challenger.

5.1 Generic Attack

We present in this subsection a generic attack against original and modified CCM, assuming only that the nonces may collide. The complexity of this attack is $\mathcal{O}(2^\ell + 2^{(t+\ell)/2})$ encryption queries and one verification query, where t is the tag bit size and ℓ the nonce bit size, and its success probability is very close to 1. Let M_3 be a message block and $(M_1^i, M_2^i)_i$ be a set of $2^{(t+\ell)/2}$ 2-block messages. The adversary makes the $2^{(t+\ell)/2}$ encryption queries (M_1^i, M_2^i, M_3) and receives the answer $N^i, (C_0^i, C_1^i, C_2^i, C_3^i)$. With high probability, there are two different indexes i and k such that $N^i = N^k$ and $C_0^i = C_0^k$. Since the nonces are the same, the counter values used for the encryption are the same and thus, since the encryptions of the tags are the same, there is a collision between the two tags. Since the last block of the two messages are the same, it means that the collision appears in fact before the last block and thus $CBC(B_0 \| M_1^i \| M_2^i \| M_3) = CBC(B_0 \| M_1^k \| M_2^k \| M_3)$ for any n -bit block M_3' (note that the collision between the nonces implies a collision between the B_0 values since the plaintexts are of the same octet length). Let $M_3' \neq M_3$ and let repeat $\mathcal{O}(2^\ell)$ times the encryption query $M_1^i \| M_2^i \| M_3'$ until the answer involves the nonce N^i . Let denote C_0, C_1, C_2, C_3 the corresponding ciphertext, and let $C'_0 = C_0, C'_1 = C_1 \oplus M_1^i \oplus M_1^k, C'_2 = C_2 \oplus M_2^i \oplus M_2^k, \text{ and } C'_3 = C_3$. The ciphertext $N^k, (C'_0, C'_1, C'_2, C'_3)$ is valid for the message $M_1^k \| M_2^k \| M_3'$ (message which was not previously asked).

Note that in the case when $t \leq \ell$ this attack requires $\mathcal{O}(2^\ell)$ encryption queries, 1 verification query, all of at most 3 blocks. Therefore if theorem 4 would apply, the success probability of such an adversary would be upper bounded by $2^{9-n} + 2^{-t}$, whereas it is nearly equal to 1. If this attack is not practical, it illustrates the fact that allowing random nonces strongly decreases the security of CCM.

5.2 If the Data Length was Not Precised

In previous attack we have relieved the constraint that nonces should not collide. In the two following attacks, we still allow the nonces to collide and in addition we assume that the formatted inputs do not form a prefix-free set. We show that, in this case, the attacks can be even worse than the previous one.

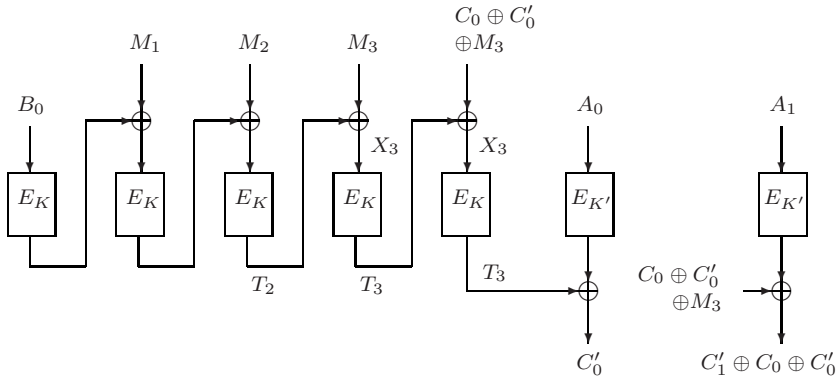


Fig. 3. Forgery attempt: $C_0 \oplus C'_0$ equals $T_2 \oplus T_3$

We remind that in CCM, B_0 depends on the message length but not of the associated data length (however if there are associated data, a flag bit is set to 1). If there are associated data, their length is encoded at most on the 10 first blocks of B . Since the length of the associated data must be known to authenticate, the authentication cannot be done online and this reduce CCM efficiency. For the following attack we retrieve this constraint, assuming that the associated data length is not concatenated *before* the associated data themselves (it could be encoded after the associated data even if for simplicity we just skip it). The attacker will use this remark to forge a valid ciphertext from $2^{\ell/2}$ encryption queries.

We assume for the attack that associated data are always used, so that the flag bit used in B_0 is always set to 1. This attack exploits some relations he can deduce from the collision on the nonce values: the attacker queries the encryption oracle for messages of two blocks and three blocks. In both cases, only the last block is encrypted. He thus collects ciphertext $(N_i, C_0^i || C_1^i)$ corresponding to the authentication of $M_1 || M_2$ and the encryption of M_2 under different nonces, and ciphertexts $(N_j^i, C_0^j || C_1^j)$ corresponding to the authentication of $M_1 || M_2 || M_3$ and the encryption of M_3 under different nonces. By the birthday paradox, after $2^{\ell/2}$ queries, there is a collision between two nonces, one used for a query in the first set and the other for a query in the second set. Thus, there exists i and j such that $N_i = N_j$. Since plaintexts to authenticate and encrypt are of the same length in both cases, we also have $B_0^i = B_0^j$. For simplicity, the corresponding ciphertexts are denoted $(N, C_0 || C_1)$ and $(N, C'_0 || C'_1 || C'_2)$.

The attacker can now compute the value

$$C_0 \oplus C'_0 = T \oplus T'$$

where T is the CBC-MAC value computed for the first message and T' is the CBC-MAC value for the second. He thus can forge a valid ciphertext for the message $M_1 || M_2 || M_3 || M_3 \oplus C_0 \oplus C'_0$ where only the last block is encrypted. The corresponding ciphertext is thus $(N, C'_0 || C'_1 \oplus C_0 \oplus C'_0)$ with the associated data $M_1 || M_2 || M_3$. Figure 3 resumes this forgery.

The complexity of the attack is $\mathcal{O}(2^{\ell/2})$ encryption queries. As $\ell \leq n$, it means that relieve the constraint on the format of the authenticated data would lead to better attacks than the birthday paradox bound proved by Jonsson.

5.3 Random Nonces with Inputs Not Formatting

Finally, in next attack we consider the case where the nonces are random and the message length is not put in the pre-initial formatted input B_0 . This way CCM computation can be done online, but however we show that in this case, even without additional authenticated data, the security of CCM decreases.

The attacker will make 3 kinds of queries: the first ones are composed with a plaintext of a single block denoted M_1 . This block is the same for all the queries. The second kind of queries is composed with messages of two blocks, $M_1||M_2$ where M_1 is the same block as the one chosen for the first queries. Finally, in the third set of queries, messages are composed with 3 blocks $M_1||M_2||M_3$ where here again M_1 and M_2 are the same as before. The attackers queries the encryption oracle for these messages until a collision occurs between the nonces used for one message in each set. Thus, there exists integers i, j, k such that : $N_i = N_j = N_k$ and thus $B_0^i = B_0^j = B_0^k$ and $A_0^i = A_0^j = A_0^k$ (since B_0 does not depend on the message length anymore). The attacker is now able to forge a valid ciphertext. We denote by $(N, C_0^1||C_1^1)$ the corresponding ciphertext for the one block message, $(N, C_0^2||C_1^2||C_2^2)$ the ciphertext for the two blocks message and $(N, C_0^3||C_1^3||C_2^3||C_3^3)$ the ciphertext for the three blocks message. Due to the choice of the message blocks M_1 and M_2 and since the nonces collide for these three encryptions, we remark that $C_1^1 = C_1^2 = C_1^3$ and $C_2^2 = C_2^3$. The first ones are briefly denoted by C_1 and the second by C_2 . We also denote by T_1 the CBC-MAC value for M_1 , T_2 the one for $M_1||M_2$, and T_3 the one for $M_1||M_2||M_3$. These notations are given in figure □.

Due to the collision between the nonces used for these three encryptions, the value $C_0^1 \oplus C_0^2$ is equal to $T_1 \oplus T_2$. Although the attackers does not know the values T_1 and T_2 , he can exploit the knowledge of their bit-wise addition to forge a valid ciphertext : indeed, the ciphertext $C_0^3||C_1||C_2 \oplus M_2 \oplus C_0^1 \oplus C_0^2 \oplus M_3$ is valid for the plaintext $M_1||C_0^1 \oplus C_0^2 \oplus M_3$ and the nonce N . The input to the third encryption box in the CBC chain is the bit-wise addition of the plaintext block $C_0^1 \oplus C_0^2 \oplus M_3$ and the previous output T_1 . Since $C_0^1 \oplus C_0^2 = T_1 \oplus T_2$, the input is just $T_2 \oplus M_3$, that is to say the input to the fourth encryption box in the CBC for the encryption of $M_1||M_2||M_3$. Thus, the output is T_3 (unknown to the adversary) and the first ciphertext block is C_0^3 . The second ciphertext block is easily computable from C_2 due to the malleability of the counter mode.

We now estimate the complexity of this attack and in particular the average number of queries needed. The attackers queries the encryption oracle until a collision appears between the nonces for three of them. By the birthday paradox, a collision occurs for two encryption queries when $2^{\ell/2}$ nonces have been chosen at random. After $2^{2\ell/3}$ queries for M_1 and $M_1||M_2$, there are in average $2^{4\ell/3}/2^\ell$ collisions, *i.e.* $2^{\ell/3}$ pairs of ciphertexts computed with the same nonce. If we consider this set of ciphertext pairs and the set of $2^{2\ell/3}$ ciphertexts for $M_1||M_2||M_3$,

there are in average triplet of ciphertexts computed with the same nonce. In a general case, if S_i is the number of ciphertext in each set, there is a 3-collision on the nonces used if and only if $S_1 \times S_2 \times S_3$ equals 2^ℓ . Choosing $S_i = 2^{2\ell/3}$ is the best compromise. Finally the attacker can forge a valid ciphertext with the help of $3 \times 2^{2\ell/3}$ encryption queries. If $\ell \leq 96 = 3n/4$ then $2\ell/3 \leq n/2$ and this attack is better than the security bound given by Jonssoon.

This attack illustrates the fact that if one wants to preserve the security, one cannot increase CCM efficiency removing the particular format, with the message length appended to the beginning of the message, and allowing repeating nonces.

6 Conclusion

In this paper we have studied the security of the CCM authenticated encryption scheme and of a modified version. We have shown that slightly modifying CCM one can prove the $\mathcal{O}(2^n)$ security for authenticity, security which only conjectures for CCM. Additionally, the modified version of CCM provably guarantees an optimal security for privacy.

Besides, we have studied the CCM (and also modified CCM) properties that restrict its efficiency. We exhibit that if we relieve some of them (if we let nonces collide, if we break the prefix-freeness of authenticated messages removing the message and/or authenticated data length) then one can mount attacks which are better than the expected or proved security bound by Jonssoon.

References

1. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, October 1997, pp. 394–403. IEEE Computer Society Press, Los Alamitos (1997)
3. Bellare, M., Goldreich, O., Mityagin, A.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. Eprint cryptology archive 2004/309 (2004), <http://eprint.iacr.org>
4. Bellare, M., Impagliazzo, R.: A Tool for Obtaining Tighter Security Analyses of Pseudorandom Function Based Constructions, With Applications to PRF-PRP conversion. Cryptology ePrint archive, Report 1999/024, <http://eprint.iacr.org>
5. Bellare, M., Namprempe, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
6. Bellare, M., Pietrzak, K., Rogaway, P.: Improved security analyses for CBC MACs. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 527–545. Springer, Heidelberg (2005)
7. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer, Heidelberg (2004)

8. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004)
9. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. In: Proc. of the 23rd STOC, ACM Press, New York (1991)
10. Dworkin, N.M.: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, NIST Special Publication 800-38C (May 2002)
11. Hall, C., Wagner, D., Kelsey, J., Schneier, B.: Building PRFs from PRPs. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 370–389. Springer, Heidelberg (1998)
12. Jonsson, J.: On the security of CTR + CBC-MAC. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 76–93. Springer, Heidelberg (2003)
13. Jutla, C.: Encryption Modes with Almost Free Message Integrity. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 529–544. Springer, Heidelberg (2001)
14. Katz, J., Yung, M.: Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology* 19(1), 67–95 (2006)
15. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
16. Lucks, S.: The Sum of PRP is a Secure PRF. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 470–484. Springer, Heidelberg (2000)
17. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS 2001, pp. 196–205. ACM Press, New York (November 2001)
18. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: Proceedings of the 8th Conference on Computer and Communications Security, pp. 196–205. ACM Press, New York (2001)
19. Rogaway, P., Wagner, D.: A Critique of CCM, Eprint cryptology archive 2003/070 (February 2003), <http://eprint.iacr.org>
20. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004)
21. Special Publication, N.: 800-38C. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality (May 2004), <http://csrc.nist.gov/CryptoToolkit/modes/>
22. Whiting, D., Housley, R., Ferguson, N.: IEEE 802.11-02/001r2: AES Encryption and Authentication Using CTR Mode and CBC-MAC (March 2002)

A CCM Encryption and Decryption Algorithms

In this section we give a description of CCM encryption and decryption algorithm in pseudo-code. For the notations see subsection 3.1. We remind that the A_i , the inputs for the counter mode, are derived from the nonce N , and the B_i are derived from the size of the message, the size of the associated data, the associated data itself, and the message itself.

Algorithm 1. CCM Encryption(M, D)

```

1: Choose  $N$ ,
2: function AUTHENTICATION( $N, D, M$ )
3:   Generates  $B_0, \dots, B_r$  from  $N, M$  and  $D$ .
4:    $X_0 \leftarrow E(B_0)$ 
5:   for  $i \leftarrow 1, r$  do
6:      $X_i \leftarrow E(B_i \oplus X_{i-1})$ 
7:   end for
8:    $T \leftarrow \text{lsb}_t(X_r)$ 
9:   return  $T$ 
10: end function
11: function ENCRYPTION( $N, D, T, M$ )
12:   Generates  $A_0, \dots, A_m$  from  $N$ .
13:    $c_0 \leftarrow \text{lsb}_t(E(A_0)) \oplus T$ 
14:   for  $i \leftarrow 1, m$  do
15:      $C_i \leftarrow E(A_i) \oplus M_i$ 
16:   end for
17:    $C \leftarrow C_0, \dots, C_m$ 
18:   return ( $N, C, D$ )
19: end function

```

Algorithm 2. CCM Decryption(N, C, D)

```

1: Generates  $A_0, \dots, A_m$  from  $N$ .
2:  $T \leftarrow \text{lsb}_t(E(A_0)) \oplus C_0$ 
3: for  $i \leftarrow 1, m$  do
4:    $M_i \leftarrow E(A_i) \oplus C_i$ 
5: end for
6:  $M \leftarrow M_0, \dots, M_m$ 
7: Generates  $B_0, \dots, B_r$  from  $N, M$  and  $D$ .
8:  $X_0 \leftarrow E(B_0)$ 
9: for  $i \leftarrow 1, r$  do
10:   $X_i \leftarrow E(B_i \oplus X_{i-1})$ 
11: end for
12: if  $T == \text{lsb}_t(X_r)$  then return ( $M, D$ )
13: else return  $\perp$ 
14: end if

```

$wNAF^*$, an Efficient Left-to-Right Signed Digit Recoding Algorithm

Brian King

Indiana University Purdue University Indianapolis
briking@iupui.edu

Abstract. Efficient implementations of cryptosystems are important in order to conserve resources, memory, power, etc., which will enable resource-limited devices to compute necessary cryptographic operations. One technique that successfully reduces the number of necessary operations is the use of a signed digit representation for the key, because it reduces the nonzero density of the representation. One such signed digit representation is the *non-adjacent form* or *NAF*. Moreover, one can make more reductions in the number of nonzero symbols of the key by expressing the key with a w -ary *NAF* or $wNAF$ form. A drawback is that one needs to parse the key twice, once to construct the $wNAF$ representation and the second time to perform the necessary cryptographic operation. At Crypto 2004 [10], Okeya et. al. introduced a w -ary representation $wMOF$, which possess the same nonzero density as $wNAF$, as well as an algorithm that computes $wMOF$ in a left-to-right manner utilizing very little memory (“memory-less”). At that time, the authors noted that a left-to-right “memory-less” algorithm that computes $wNAF$ is an open problem. In this work, we define $wNAF^*$, a generalization of $wNAF$. Further, we construct a left-to-right “memory-less” algorithm that computes the w -ary $wNAF^*$ representation of a key and demonstrate that $wNAF^*$ is as efficient as $wNAF$. Our work will demonstrate that the left-to-right $wNAF^*$ recoding algorithm closely resembles the right-to-left $wNAF$ recoding algorithm.

1 Introduction

It is well known that if one uses a signed digit representation of the cryptographic key then one can reduce the number of complex computations that are needed to perform the cryptographic operation. One such signed digit representation is the *non-adjacent form* or *NAF*. By expressing a key in NAF form, one will reduce the number of nonzero symbols, however this will introduce signed digits. Fortunately, in many algebraic settings an inverse is very efficient to compute. This is especially true when using elliptic curve cryptosystems (ECC), where the inverse (negative) of an ECC point can be trivially computed from the EC point. When using the NAF form of a key, the computational complexity of computing the scalar multiple kP will be very efficient, since the hamming weight of a key in NAF form is less than the hamming weight of a key. One can make further

reductions in the number of nonzero symbols of the key by expressing it with a w -ary NAF or w NAF form [2]. RSA and discrete-log cryptosystems would also benefit by expressing a key in NAF or w NAF form because of the reduction of the hamming weight (number of nonzero bits) of the key. In order to be more succinct, we will emphasize the use of NAF and w NAF with ECC, however our work will impact efficient implementations of all cryptosystems.

Algorithms demonstrating how to compute the NAF form of a key can be found in [2,12,4] and algorithms to compute the w NAF form are provided in [2,4]. In addition to NAF, there are other alternative ways to reduce the number of nonzero symbols in the key representation by using techniques such as *sliding window with NAF* [3].

At Crypto 2004 Okeya et. al. [10] introduced a left-to-right binary signed digit recoding algorithm called Mutual Opposite Form or *MOF*. The goal of the authors of [10] was to develop an algorithm that constructs an “efficient signed digit representation that could be computed in a left-to-right manner without requiring additional storage”. That is, a signed digit representation that can be computed in a left-to-right manner on-the-fly. The importance of such a construction is that as the key bits are generated (in a left-to-right manner), one can immediately construct the signed digit representation. Further, one can immediately start the computation of the cryptographic operations as the most significant digits of the signed digit representation become available. This is a good security practice, in that the secret key should only be available to software modules that require it. Moreover, it is good practice to limit the number of software modules that use/possess key. Consequently, if the cryptographic computation can be computed simultaneously as the signed digits are generated, this would limit the access of the key to software modules.

A left-to-right construction of a signed digit representation NAF was first constructed in [5], however this construction was limited to $w = 2$ and the authors did not provide a left-to-right algorithm of the more efficient representation w NAF. In [10], it was noted that the construction of a memory-less left-to-right algorithm for w NAF is an open problem. In this work, we examine the open problem posed in [10], and provide an algorithm that can construct a left-to-right w NAF* algorithm, where w NAF* is a left-to-right generalization of w NAF.

Summary of our results. We will address an open problem by defining a signed digit representation w NAF* and constructing a left-to-right “memory-less” method for computing w NAF*. Our work will establish the relationship between left-to-right w -ary signed digit recoding and right-to-left w -ary signed digit recoding. We will also provide several algorithms, including an algorithm that computes the ECC scalar multiple kP , processing the key k using the left-to-right w NAF* recoding.

2 Background

In [11], Reitwiesner introduced the signed digit representation referred to as NAF. The definition that an integer k is written in NAF form is:

Definition 1 (NAF). [2] A nonnegative integer $k = \sum_{i=0}^{n-1} k_i 2^i$ where $k_i \in \{-1, 0, 1\}$ is said to be in non-adjacent form (NAF) provided $k_i \cdot k_{i+1} = 0$ for $i = 0, \dots, n - 2$.

Reitwiesner’s algorithm [11] efficiently converted a binary number to signed-digit NAF form using a right-to-left method, see Algorithm 1.

Throughout this paper we will abbreviate -1 as $\bar{1}$. The following are some well-known results concerning the NAF form of an integer. Every positive integer k has a unique NAF representation [1, 2]. The length of k written in NAF-form is at most one bit longer than the length of k . In [1], it was proven that the expected number of nonzero symbols in a NAF representation was $1/3$ times the length of k . In general, one would expect a random k to have an equal number of 1’s as 0’s.

Joye and Yen proposed two left-to-right binary signed-digit recoding algorithms in [5]. Based on Reitwiesner’s algorithm and the left-to-right addition algorithm [9], Joye and Yen developed the first left-to-right recoding algorithm which preserves the NAF property. Joye and Yen also developed a more efficient left-to-right signed digit recoding algorithm whose output possessed the same nonzero density properties as a NAF representation.

One can make further reductions in the number of nonzero symbols by utilizing a w -ary NAF form or $wNAF$. Formally the definition of $wNAF$ is:

Definition 2 (wNAF). [4] Let $w > 2$ and k a positive integer. We say that $k = \sum_{i=0}^{n-1} k_i 2^i$ is a $wNAF$ representation of integer k provided (i) $k_{n-1} \neq 0$, (ii) for all nonzero k_i , k_i is an odd integer with $|k_i| < 2^{w-1}$, and (iii) at most one of any w consecutive digits $k_i, k_{i+1}, \dots, k_{i+w-1}$ is nonzero.

The term 2NAF is often used to describe a NAF representation. The ratio of nonzero symbols to symbols in a wNAF representation has been shown to be on average $1/(w + 1)$ [2, 10].

At Crypto 2004, Okeya, et.al. [10] created a sparse representation of a key, which can be constructed using a left-to-right pass through the key, utilizing little memory. The representation they introduced was the *Mutual Opposite Form (MOF)*.

Definition 3 (MOF). [10] A n -bit integer is represented using the mutual opposite form (MOF) provided:

1. the signs of adjacent nonzero bits are opposite sign
2. the most significant bit is 1 and the least significant bit is -1 , unless all bits are zero

Okeya et. al. [10] then generalized this to construct wMOF.

Definition 4 (wMOF). [10] A signed digit representation satisfies wMOF provided

1. The most significant nonzero digit is positive.
2. All but the least significant nonzero digit x are adjoint by $w - 1$ zeros as (i) if $2^{k-1} < |x| < 2^k$ for $2 \leq k \leq w - 1$ the pattern is $0\dots 0x0\dots 0$ (k leading zeros and $w - k - 1$ trailing zeros),

(ii) if $|x| = 1$ we have either the pattern $x000\dots0$ ($w - 1$ trailing zeros) and the next lower nonzero digit has opposite sign to x or the pattern $0x0\dots0$ ($w - 2$ trailing zeros) and the next lower digit has the same sign as x , and

(iii) if x is the least significant nonzero digit, it is possible that the number of right-hand adjacent zeros is smaller than the stated above. It is not possible that the last nonzero digit is a 1 following any nonzero digit.

3. Each nonzero digit is odd and less than 2^{w-1} in absolute value.

In [10], the authors provided an algorithm that constructs the $wMOF$ representation in a left-to-right manner and showed that the nonzero density of a $wMOF$ representation was $1/(w + 1)$, the same as $wNAF$.

The amount of precomputations represent both a computational resource requirement as well as a memory requirement. Because the inverse (negative) of the group operation in an Elliptic Curve Cryptosystem is trivial to compute, one does not need to pre-compute nor store negative multiples of the EC point, since the negative of an EC point can be computed requiring very little resources and can be computed as needed. In addition to the left-to-right algorithm which computes the $wMOF$ representation, Okeya et. al. also constructed a left-to-right algorithm that computes $wNAF$. Unfortunately their algorithm requires additional memory and hence it is not a memoryless left-to-right algorithm. In their left-to-right $wNAF$ algorithm, they require $O(\frac{n}{w})$ memory (see Table 5). In practice since w will be constant, this is $O(n)$ amount of memory. In our algorithm that computes $wNAF^*$, we require no additional memory. The nonzero density for $wNAF$ is the optimal value $1/(w + 1)$. We will establish that the nonzero density of $wNAF^*$ is also $\frac{1}{w+1}$ and the precomputations needed by $wNAF^*$ is identical to $wNAF$. Reminder, other cryptosystems like RSA and discrete-log cryptosystems would realize improved efficiency when using a key in $wNAF$ form.

3 NAF and NAF^*

Algorithm 1 illustrates Reitweisner’s canonical recoding of the key k , computed in a right to left manner. In this case $k = (d_{n-1}, d_{n-2}, \dots, d_2, d_1, d_0) = \sum_{j=0}^{n-1} d_j \cdot 2^j$. This algorithm saves each carry c_j that is produced during each iteration, though one only needs to know the current carry-in, the current symbol d_i and the lookahead symbol d_{i+1} .

Algorithm 1 can be expressed in a table format, see Table 1. Here d_i denotes the current symbol, d_{i+1} denotes the lookahead symbol and C denotes the carry-in. The output symbol is δ_i . A second output symbol is C which is the carry-out and becomes the next carry-in symbol.

In addition to constructing a left-to-right $wMOF$ representation, Okeya, et. al. constructed a left-to-right NAF algorithm (see Algorithm 5 in [10]) but this construction required external memory. They needed to track the number of consecutive ones that are visited. Because the number of consecutive ones could be on the order of n , at least $O(\log_2 n)$ memory is needed. By applying a innovated technique we can create a left-to-right binary encoding which we call NAF^* , which

Algorithm 1. Reitwiesner’s canonical recoding [5,2,11]

```

1: INPUT:  $d_n, \dots, d_0$ 
2: OUTPUT:  $\delta_{n+1}, \dots, \delta_0$ 
3:  $c_0 \leftarrow 0$ 
4: for  $j = 0$  to  $n + 1$  do
5:    $c_{j+1} \leftarrow \lfloor (d_j + d_{j+1} + c_j)/2 \rfloor$ 
6:    $\delta_j \leftarrow d_j + c_j - 2c_{j+1}$ 
7: return  $\delta_{n+1}\delta_{n-1} \dots \delta_0$ 
    
```

Table 1. Reitwiesner’s algorithm to compute right-to-left NAF

Carry-in C	Current Symbol d_i	Lookahead Symbol d_{i+1}	Output result δ_i
0	0	x	$\delta_i = 0, i \leftarrow i + 1,$ and set $C = 0$
0	1	0	$\delta_i = 1, i \leftarrow i + 1,$ and set $C = 0$
0	1	1	$\delta_i = -1,$ set $C = 1$ and $i \leftarrow i + 1$
1	0	0	$\delta_i = 1, i \leftarrow i + 1,$ and set $C = 0$
1	0	1	$\delta_i = -1$ and set $C = 1$ and $i \leftarrow i + 1$
1	1	x	$\delta_i = 0$ and set $C = 1, i \leftarrow i + 1$

Here x

represents a don’t care, it can be either 0 or 1.

requires no additional storage. We amend the definition of NAF to form our definition of NAF^* . Our technique mimics the right-to-left NAF computation. For example when encountering a sequence 11 (in a right-to-left manner) the NAF computation makes the replacement of 11 with $0\bar{1}$ and a left carry of one. When we encounter 11 in a left-to-right manner, then we have already processed the bit prior to the sequence 11 (to the left). Since we have already processed the bits to the left, we can only create “carries to the right”. Hence we replace 11 by 20 with a carry-to-the right of $\bar{2}$ (here $\bar{2}$ denotes negative 2). Of course the sequence 11 represents $3=2*1+1$, whereas the sequence 20 with a carry-to-the right of $\bar{2}$ represents $2*2+1*0+\frac{1}{2}*\bar{2}=3$. It is of course straightforward to handle sequences 01, 10, and 00. What remains to be considered are the cases when we have carry-in’s from the left, which can occur, such as in the case described above. The only two possible carry-in symbols will be 0 and $-2 = \bar{2}$. Recall, we are examining current symbol d_i while the lookahead symbol is d_{i-1} . The possible values that these two consecutive symbols can represent are determined by $2*d_i + d_{i-1}$ (integers between 0 and 3). If we have a carry-in (from the left) of -2 , then this would be placed in the d_i place. This would be equivalent to a value of $-2*2 = -4$, thus we must add -4 to the possible values, resulting in a sequence of integers between -4 to -1 . In NAF^* , we use an extended set of symbols that consist of 0, ± 1 , and ± 2 . Table 2 illustrates how we handle the cases of a left carry-in of $-2 = \bar{2}$. The definition of NAF^* is identical to that of NAF except the symbols can consist of 0, ± 1 , and ± 2 (whereas NAF restricts them to 0 and ± 1). An integer $k = \sum_{i=0}^{n-1} k_i 2^i$ is expressed in NAF^* form provided $k_i \in \{0, \pm 1, \pm 2\}$ and $k_i \cdot k_{i-1} = 0$ for all $i = 1, \dots, n - 1$.

The computation of the NAF^* recoded key can be generated using Table 2, where the initial value for C is 0 and $i = n - 1$. Here the key $k = d_{n-1} \dots d_1 d_0$. Further, for obvious reasons we use $d_{-1} = d_{-2} = 0$.

Table 2. Left-to-right NAF^*

Carry-in C	Current Symbol d_i	LookAhead Symbol d_{i-1}	result δ_i
0	0	x	$\delta_i = 0, i \leftarrow i - 1$, and set $C = 0$
0	1	0	$\delta_i = 1, i \leftarrow i - 1$, and set $C = 0$
0	1	1	$\delta_i = 2, i \leftarrow i - 1$, and set $C = -2$
-2	0	0	$\delta_i = -2, i \leftarrow i - 1$, and set $C = 0$
-2	0	1	$\delta_i = -1, i \leftarrow i - 1$ and set $C = -2$
-2	1	x	$\delta_i = 0, i \leftarrow i - 1$ and set $C = -2$

Here x represents a *don't care*, it can be either 0 or 1.

We now describe NAF^* in algorithm form. That is the following algorithm is an implementation of Table 2.

Algorithm 2. NAF^*

- 1: **INPUT:** d_n, \dots, d_0
 - 2: **OUTPUT:** $\delta_n, \dots, \delta_0 \delta_{-1}$
 - 3: $c_n \leftarrow 0$
 - 4: **for** $j = n$ **downto** 0 **do**
 - 5: $c_{j-1} \leftarrow -2 \cdot ((d_j + d_{j-1} + \frac{|c_j|}{2})/2)$
 - 6: $\delta_j \leftarrow d_j + c_j - \frac{1}{2} \cdot c_{j-1}$
 - 7: $\delta_{-1} \leftarrow c_{-1}$
 - 8: **return** $\delta_n \delta_{n-1} \dots \delta_0 \delta_{-1}$
-

In Algorithm 2, because we execute left-to-right, carries are either -2 or 0 . This explains the use of the absolute value and the fraction $\frac{1}{2}$ in line 5 and the use of the fraction $\frac{1}{2}$ in line 6.

Example 1. The following example illustrates both the NAF recoding and the NAF^* recoding for a key k .

Key k	1 0 0 1 1 0 1 1 0 1 0 1 1 1 1	
NAF recoding of k	1 0 1 0 0 $\bar{1}$ 0 0 $\bar{1}$ 0 $\bar{1}$ 0 0 0 $\bar{1}$	
NAF^* recoding of k	1 0 0 2 0 $\bar{1}$ 0 0 $\bar{1}$ 0 $\bar{1}$ 0 0 0 0	$\bar{2}$

The reason for the $\bar{2}$ symbol in the NAF^* recoding (see the far right symbol), is that this occurs in the case where $i = -1$, i.e. $\delta_{-1} = \bar{2}$. That is, the length of the NAF^* recoded key is one symbol longer than that of the key $k = d_{n-1}, \dots, d_1, d_0$. Much like NAF, the recoding NAF^* may have a length of one symbol longer than the length of the key but the extra symbol occurs in the far right place (where $i = -1$).

Theorem 1. *The NAF^* algorithm, as described by Table 2, when applied to k will produce a recoded sequence of symbols satisfying NAF^* , such that the sequence is equivalent to k*

Proof. To establish this theorem we are left to show that after each iteration i of applying Table 2, we have a sequence of symbols which is equivalent to the key k . We will assume that prior to the i^{th} iteration the result computes the key. Prior to the i^{th} iteration we have completed the $i + 1^{st}$ iteration. Thus we have computed $\delta_{n-1}, \dots, \delta_{i+1}$, the carry C , which we will denote as C_{IN} (since this is the carry-in to the i^{th} iteration), as well as the symbols d_i, \dots, d_0 which have not been processed. Thus

$$k = C_{IN} \cdot 2^i + \sum_{j=i+1}^{n-1} \delta_j \cdot 2^j + \sum_{j=0}^i d_j \cdot 2^j. \tag{1}$$

Now after the i^{th} iteration we have processed d_i based on C_{IN} and d_{i-1} . The result is that we have now computed δ_i and C_{OUT} (the C value which is the carry-out after the i^{th} iteration) based on Table 2. Consider $C_{OUT} \cdot 2^{i-1} + \delta_i \cdot 2^i + \sum_{j=i+1}^{n-1} \delta_j \cdot 2^j + \sum_{j=0}^{i-1} d_j \cdot 2^j$. We need to show that this value is equivalent to equation (1). By cancelling the common terms, we are left to show $C_{IN} \cdot 2 + d_i \cdot 2 = C_{OUT} + \delta_i \cdot 2$. By examining each row of Table 2, we see that this equation is valid for each possible input and so this establishes the theorem. •

Observe that there is a natural mapping between the execution of the NAF algorithm and the execution of the NAF^* algorithm. This is clearly demonstrated by a comparison of Table 1 and Table 2. The mapping is implied by the correspondence between rows of the tables. This is a one-to-one mapping such that NAF outputs a nonzero symbol iff NAF^* outputs a nonzero symbol and NAF outputs a carry term that is nonzero iff NAF^* outputs a carry term that is nonzero. By [1], the expected hamming weight of a key, of length n , recoded in NAF is $\frac{n}{3}$. This mapping, together with the result by [1], establishes the following.

Theorem 2. *The expected hamming weight of a NAF^* recoded key of length n is $\frac{n}{3}$.*

Proof. This theorem follows from the result that the expected hamming weight of a key recoded in NAF form is $\frac{n}{3}$, and the fact that there exists a natural one-to-one mapping between NAF recoding and a NAF^* recoding (as described above). Since the input distribution, as well as the distribution of zero vs. nonzero

carry symbols are identical, we see that under this mapping and due to the result by [1], the expected hamming weight of a NAF^* recoded key is $\frac{n}{3}$. •

Consequently NAF^* is as “efficient” as NAF except NAF^* allows the use of the symbols 2 and -2 . Thus when applying NAF^* to perform an ECC scalar multiple, it appears that a precomputation of $2P$ will need to be performed, however $2P$ always must be computed. Consequently, no additional precomputation is needed. The ECC scalar multiple algorithm is provided in the Appendix, see Algorithm 4.

4 $wNAF$ and $wNAF^*$

The following is a right-to-left calculation for $wNAF$ as provided in [12,10]. Note that the term “mods”, as used in Algorithm 2, is defined as: $x \bmod y$ returns a value j such that $-y/2 \leq j < y/2$ and $j \bmod y = x \bmod y$. For example $5 \bmod 16 = 5$ and $11 \bmod 16 = -5$.

Algorithm 3. $wNAF$ [12,10]

```

1: Input: width  $w$  and  $n$ -bit integer  $d$ 
2: Output:  $wNAF$   $\delta_n \delta_{n-1} \dots \delta_0$  of  $d$ 
3:  $i \leftarrow 0$ 
4: while  $d \geq 1$  do
5:   if  $d$  is even then
6:      $\delta_i \leftarrow 0$ 
7:   else
8:      $\delta_i \leftarrow d \bmod 2^w$ 
9:      $d \leftarrow d - \delta_i$ 
10:     $d \leftarrow d/2$ ;  $i \leftarrow i + 1$ 
11: return  $\delta_n \delta_{n-1} \dots \delta_0$ 

```

Algorithm 3, which computes the $wNAF$ recoding, can be rewritten in table form, see Table 3. The correctness of the table can be established by considering the various cases and applying the above algorithm. We omit the proof of correctness for the table.

4.1 $wNAF^*$

We modify the NAF^* construction and apply techniques that mimic a right-to-left $wNAF$ construction to construct a $wNAF^*$ definition and recoding algorithm. Again as our algorithm parses the key in a left-to-right fashion we will restrict ourselves to a carry-in of 0 or -2 . Observe that in the definition of $wNAF$, a nonzero integer k_i is such that it is a most $w - 1$ bits, where the low-bit is a one (i.e. it is an odd integer) and the high bit is zero (thus the absolute

Table 3. Right-to-left $wNAF$

Carry-in C	Input $R = (d_{i+w-1}, \dots, d_{i+1}, d_i)$	Output δ_i result
0	$d_i = 0$	$\delta_i = 0, i \leftarrow i + 1, \text{ and } C = 0$
0	$d_i = 1 \text{ and } d_{i+w-1} = 0$	$(\delta_{i+w-1}, \dots, \delta_{i+1}, \delta_i) = (0, 0, \dots, 0, R)$ $i \leftarrow i + w, \text{ and } C = 0$
0	$d_i = 1 \text{ and } d_{i+w-1} = 1 \exists j d_j = 0$	$(\delta_{i+w-1}, \dots, \delta_{i+1}, \delta_i) = (0, 0, \dots, 0, -(2^w - R))$ $i \leftarrow i + w, C = 1$
0	$d_i = 1 = d_{i+1} = \dots = d_{i+w-1} = 1$	$(\delta_{i+w-1}, \dots, \delta_{i+1}, \delta_i) = (0, 0, \dots, 0, -1)$ $i \leftarrow i + w, C = 1$
1	$d_i = 1$	$\delta_i = 0 \ i \leftarrow i + 1, C = 1$
1	$d_i = 0 \text{ and } d_{i+w-1} = 0$	$(\delta_{i+w-1}, \dots, \delta_{i+1}, \delta_i) = (0, 0, \dots, 0, R + 1)$ $i \leftarrow i + w, \text{ and } C = 0$
1	$d_i = 0 \text{ and } d_{i+w-1} = 1$	$(\delta_{i+w-1}, \dots, \delta_{i+1}, \delta_i) = (0, 0, \dots, 0, -(2^w - (R + 1)))$ $i \leftarrow i + w, C = 1$

value of k_i is less than 2^{w-1}). We need to provide a recoding definition for our left-to-right construction that mimics these properties. In our definition, if the symbol is nonzero the left-most bit must be nonzero and the rightmost w^{th} bit will be zero. We now formalize the definition of $wNAF^*$

Definition 5. $wNAF^*$ Let $w > 2$ and k a positive integer. Then we say that $\sum_{i=-1}^{n-1} a_i 2^i$ is a $wNAF^*$ representation of k provided $k = \sum_{i=-1}^{n-1} a_i 2^i$ and

- (i) a_i is a rational number for all i ,
- (ii) at most one of any w consecutive symbols $a_i, a_{i+1}, \dots, a_{i+w-1}$ is nonzero,
- (iii) for all i , if a_i is nonzero then $1 \leq |a_i| \leq 2$,
- (iv) for all i , the product $a_i \cdot 2^{w-2} \in \mathbb{Z}$, and
- (v) for all i , with $-1 \leq i \leq w - 1$ we have $a_i \cdot 2^i \in \mathbb{Z}$.

First observe that if $a_{-1} \neq 0$ then $1 \leq |a_{-1}| \leq 2$ and $a_{-1} \cdot 2^{-1} \in \mathbb{Z}$. Since $1 \cdot 2^{-1} \leq |a_{-1}| \cdot 2^{-1} \leq 2 \cdot 2^{-1} = 1$ we see that $|a_{-1}| = 2$. So if $a_{-1} \neq 0$ then either $a_{-1} = -2$ or $a_{-1} = 2$.

Let us now consider a_i where $i \geq w - 2$. Suppose a_i is nonzero, without loss of generality assume that $a_i > 0$. Thus $1 \leq a_i \leq 2$. Then $a_i = 1 + \epsilon$ where $\epsilon \in [0, 1]$. Therefore we can express a_i as

$$a_i = 1 + \sum_{j=1}^{\infty} a_{i,j} \frac{1}{2^j} \text{ where } a_{i,j} \in \{0, 1\} . \tag{2}$$

Now $a_i \cdot 2^{w-2} \in \mathbb{Z}$. Consequently,

$$a_i \cdot 2^{w-2} = 1 \cdot 2^{w-2} + \sum_{j=1}^{w-2} a_{i,j} \frac{2^{w-2}}{2^j} + \sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j}$$

where $a_{i,j} \in \{0, 1\}$. Observe that $1 \cdot 2^{w-2} + \sum_{j=1}^{w-2} a_{i,j} \frac{2^{w-2}}{2^j} \in \mathbb{Z}$. Thus as $a_i \cdot 2^{w-2} \in \mathbb{Z}$ we have $\sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j}$ is an integer. Now $0 \leq \sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j} \leq 1$.

Therefore since $\sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j}$ is an integer, there are two possible cases, it is either 0 or 1.

Case 1. Suppose $\sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j} = 0$. Then $a_{i,j} = 0$ for all $j = w - 1, \dots$. So in this case $a_i = 1 + \sum_{j=1}^{w-2} a_{i,j} \frac{1}{2^j}$.

Case 2. Suppose $\sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j} = 1$. Then $a_{i,j} = 1$ for all $j = w - 1, \dots$ and $\sum_{j=w-1}^{\infty} a_{i,j} \frac{2^{w-2}}{2^j} = \frac{1}{2} + \frac{1}{4} + \dots = 1$. Thus in this case $\sum_{j=w-1}^{\infty} a_{i,j} \frac{1}{2^j} = \frac{1}{2^{w-2}}$. Recall $a_i = 1 + \sum_{j=1}^{w-2} a_{i,j} \frac{1}{2^j} + \sum_{j=w-1}^{\infty} a_{i,j} \frac{1}{2^j}$, so $a_i = 1 + \sum_{j=1}^{w-2} a_{i,j} \frac{1}{2^j} + \frac{1}{2^{w-2}}$.

We now examine this latter case, case 2, more closely. Suppose $a_i = 1 + \sum_{j=1}^{w-2} a_{i,j} \frac{1}{2^j} + \frac{1}{2^{w-2}}$. Observe that if $a_{i,j} = 1$ for all $j = 1, \dots, w - 2$, then $a_i = 2$. Suppose there exists a j with $1 \leq j \leq w - 2$ such that $a_{i,j} = 0$. Since $a_i = 1 + \sum_{j=1}^{w-2} a_{i,j} \frac{1}{2^j} + \frac{1}{2^{w-2}}$, and there exists some $a_{i,j}$ equal to zero, we can simplify this as $a_i = 1 + \sum_{j=1}^{w-2} b_{i,j} \frac{1}{2^j}$ where $b_{i,j} \in \{0, 1\}$. Thus there are three possible representations for a_i , either it is 0, ± 2 or $\pm(1 + \sum_{j=1}^{w-2} b_{i,j} \frac{1}{2^j})$ where $b_{i,j} \in \{0, 1\}$.

For all cases, when a_i is expressed using the representation given in (2), we see that $a_{i,j} = 0$ for $j = w - 1, \dots$. Further, if $a_i \neq 0$, and $a_i \neq \pm 2$, then we can express a_i as $|a_i| = (1, a_{i,1}, \dots, a_{i,w-2}, 0)$. Consequently either $a_i = 0$, $a_i = \pm 2$ or it can be interpreted as a $w - 1$ bit rational number (either positive or negative). The interpretation of a_i as a $w - 1$ bit rational number comes from $|a_i| = (1, a_{i,1}, \dots, a_{i,w-2}, 0) = 1 + \frac{a_{i,1}}{2} + \frac{a_{i,2}}{2^2} + \dots + \frac{a_{i,w-2}}{2^{w-2}}$ where $a_{i,j} \in \{0, 1\}$.

Obviously the definition of $wNAF^*$ mimics the definition of $wNAF$. That is, one can view a_i as a w -bit symbol such that if a_i is non zero, and if $a_i \neq \pm 2$ then the high bit $a_{i,w-1}$ is one (this is analogous to the case that in $wNAF$ the k_i is odd) and the far right w^{th} bit is zero. This implies that $|a_i|$ is between 1 and 2, such that $|a_i| = 1 + \frac{a_{i,1}}{2} + \frac{a_{i,2}}{2^2} + \dots + \frac{a_{i,w-2}}{2^{w-2}}$.

Let R be a $w - 1$ -bit (or less) nonzero positive integer. $R = r_j, \dots, r_1$ where $r_j = 1$ and $1 \leq j \leq w - 1$. Then we define R_{frac} as

$$R_{frac} = (1, r_{j-1}, \dots, r_1)_{frac} = \sum_{i=0}^{j-1} r_{j-i} \cdot \frac{1}{2^i} = 1 + r_{j-1} \cdot \frac{1}{2} + \dots + r_1 \cdot \frac{1}{2^{j-1}}. \quad (3)$$

4.2 $wNAF^*$ Recoding Algorithm

When computing the $wNAF$ representation of a key, we may have carry-in's from the right that are 0 or 1. In the left-to-right calculation of $wNAF^*$ we could have carry-in's (from the left) of either -2 or 0. Table 4 describes how to construct the $wNAF^*$ recoding of key k . We initialize $C = 0$ and $i = n - 1$ and process and compute each output symbol δ_i in a left-to-right manner. We will continue until $i < -1$. It is possible that the last symbol generated δ_{-1} may be nonzero.

Observe that termination is guaranteed, because $k = \sum_{j=0}^{n-1} d_j 2^j$, so we interpret input values $d_{-1} = d_{-2} = 0$ (since they do not have any assigned values, they are correctly treated as zero). Also observe that it is possible that we have an output result of δ_{-1} being nonzero (this is illustrated in the Example 2).

Table 4. Left-to-right $wNAF^*$

Carry-in C	Input $R = (d_i, d_{i-1}, \dots, d_{i-w+2}, d_{i-w+1})$	Output result
0	$d_i = 0$	$\delta_i = 0, i \leftarrow i - 1, \text{ and } C = 0$
0	$d_i = 1 \text{ and } d_{i-w+1} = 0$	$(\delta_i, \delta_{i-1}, \dots, \delta_{i-w+2}, \delta_{i-w+1}) = (R_{frac}, 0, 0, \dots, 0),$ $i \leftarrow i - w \text{ and } C = 0$
0	$d_i = 1 \text{ and } d_{i-w+1} = 1 \exists j d_j = 0$	$(\delta_i, \delta_{i-1}, \dots, \delta_{i-w+2}, \delta_{i-w+1}) = (R + 1)_{frac}, 0, 0, \dots, 0),$ $i \leftarrow i - w, C = -2$
0	$d_i = 1 = \dots = d_{i-w+1} = 1$	$(\delta_i, \delta_{i-1}, \dots, \delta_{i-w+2}, \delta_{i-w+1}) = (2, 0, \dots, 0),$ $i \leftarrow i - w, C = -2$
-2	$d_i = 1$	$\delta_i = 0, i \leftarrow i - 1, C = -2$
-2	$d_i = 0 \text{ and } d_{i-w+1} = 0$	$(\delta_i, \delta_{i-1}, \dots, \delta_{i-w+2}, \delta_{i-w+1}) = (-2^w - R)_{frac}, 0, 0, \dots, 0),$ $i \leftarrow i - w \text{ and } C = 0$
-2	$d_i = 0 \text{ and } d_{i-w+1} = 1$	$(\delta_i, \delta_{i-1}, \dots, \delta_{i-w+2}, \delta_{i-w+1}) = (-2^w - (R + 1))_{frac}, 0, 0, \dots, 0),$ $i \leftarrow i - w, C = -2$

However if $\delta_{-1} \neq 0$ then δ_{-1} is either 2 or -2 . Further $\delta_{-2} = \delta_{-3} = \dots = 0$. In Theorem 3 we show that the $wNAF^*$ recoding is correct.

Example 2. We illustrate the different recodings systems for a fixed key k . For a key $k = (1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1) = 3,918,617,943$. We express k utilizing the following representations: NAF, NAF^* , 3NAF, $3NAF^*$, 4NAF, $4NAF^*$, and we provide a generalization of $4NAF^*$ which uses integer symbols.

key k	1 1 1 0 1 0 0 1 1 0 0 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1
NAF form of k	1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1
NAF^* form of k	2 0 0 1 0 2 0 2 0 2 0 1 0 0 0 1 0 1 0 2 0 0 1 0 1 0 1 0 1 0 0 0 2
3NAF form of k	1 0 0 0 0 3 0 0 0 3 0 0 1 0 0 0 1 0 0 3 0 0 0 1 0 0 3 0 0 3 0 0 1
$3NAF^*$ form of k	2 0 0 $\frac{3}{2}$ 0 0 0 $\frac{3}{2}$ 0 0 0 1 0 0 0 $\frac{3}{2}$ 0 0 1 0 0 0 $\frac{3}{2}$ 0 0 $\frac{3}{2}$ 0 0 1 0 0 0 2
4NAF form of k	0 0 0 7 0 0 0 5 0 0 0 3 0 0 0 7 0 0 0 5 0 0 0 0 3 0 0 0 5 0 0 0 7
$4NAF^*$ form of k	$\frac{7}{4}$ 0 0 0 $\frac{5}{4}$ 0 0 0 0 $\frac{7}{4}$ 0 0 0 0 0 $\frac{5}{4}$ 0 0 0 $\frac{7}{4}$ 0 0 0 $\frac{5}{4}$ 0 0 0 $\frac{3}{2}$ 0 0 0 2
Modified version ^a $4NAF^*$ form of k	0 0 7 0 0 0 5 0 0 0 0 7 0 0 0 0 5 0 0 0 7 0 0 0 5 0 0 3 0 0 0 2

^a In this representation we express entries as integers rather than rational numbers.

Observe that we do not necessarily satisfy wNAF properties.

Theorem 3. Let $\Delta = \sum_{j=-1}^{n-1} \delta_j 2^j$ be the $wNAF^*$ recoding of key k then $\Delta = k$.

Proof. Our goal is to show that after each iteration we replace key bits d_j with symbols from the $wNAF^*$ recoding δ_j such that result after replacement still equals k . Assume that prior to the i^{th} iteration we have $C_{IN} \cdot 2^i + \sum_{j=0}^i d_j \cdot 2^j + \sum_{j=i+1}^{n-1} \delta_j \cdot 2^j = k$.

We now consider each of the seven possible cases, a case for each of the seven rows of Table 4.

First case, suppose $C_{IN} = 0$ and $d_i = 0$, then trivially after executing the i^{th} iteration the resulting replacement still equals k .

Now suppose $C_{IN} = 0$ and $d_i = 1$ and $d_{i-w+1} = 0$. Then $R = d_i \dots d_{i-w+1} = 1x \dots x0$. We replace $\sum_{j=i-w+1}^i d_j \cdot 2^j$ by $R_{frac} \cdot 2^i$ which are equal. Again trivially after executing the i^{th} iteration the result still equals k .

Now suppose $C_{IN} = 0$ and $d_i = 1$, $d_{i-w+1} = 1$ and there exists j with $i - w + 1 < j < i$ such that $d_j = 0$. Then $R = d_i \dots d_{i-w+1} = 1x \dots x01 \dots 1$. Thus $R + 1 = 1x \dots x10 \dots 0$. In this case $C_{OUT} = -2$. Now $(R + 1)_{frac} \cdot 2^i + C_{OUT} \cdot 2^{i-w} = R_{frac} \cdot 2^i + 1 \cdot \frac{2^i}{2^{-w+1}} + -2 \cdot 2^{i-w} = R_{frac} \cdot 2^i = \sum_{j=i-w+1}^i d_j \cdot 2^j$. So the use of $(R + 1)_{frac}$ with C_{OUT} of -2 will provide a valid replacement.

Now suppose $C_{IN} = 0$ and $d_i = 1 = \dots = d_{i-w+1} = 1$. Then $R = d_i \dots d_{i-w+1} = 11 \dots 111 \dots 1$. Thus $R + 1 = 2^{w+1}$. Note $(R + 1)_{frac} = 2$. In this case $C_{OUT} = -2$. With an argument similar to the above case, the use of $(R + 1)_{frac}$ with C_{OUT} of -2 will provide a valid replacement.

Now suppose $C_{IN} = -2$ and $d_i = 1$. Then $-2 + 1 = -1$ so the use of $\delta_i = 0$ and $C_{OUT} = -2$ will provide a valid replacement.

Now suppose $C_{IN} = -2$ and $d_i = 0$ and $d_{i-w+1} = 0$. Then $R = d_i \dots d_{i-w+1} = 1x \dots x0$. Thus $-(2^w - R)$ represents the sum of C_{IN} with R . The use of $-(2^w - R)_{frac}$ with C_{OUT} of 0 will provide a valid replacement.

Now suppose $C_{IN} = -2$ and $d_i = 0$ and $d_{i-w+1} = 1$. Then $R = d_i \dots d_{i-w+1} = 0x \dots x1$. Thus $R + 1 = x'x' \dots x'0$. Consequently $-(2^w - (R + 1))$ represents the sum of C_{IN} with R producing a $C_{OUT} = -2$. The use of $-(2^w - (R + 1))_{frac}$ with C_{OUT} of -2 will provide a valid replacement.

Thus all seven cases produce a valid replacement. •

Theorem 3 established that the $wNAF^*$ representation is correct (equivalent to k). We now discuss the “efficiency” of the $wNAF^*$ recoding.

Theorem 4. The average nonzero density of a $wNAF^*$ recoding is $\frac{1}{w+1}$.

Proof. There exists a natural mapping between $wNAF$ and $wNAF^*$. This can be seen by examining each row of Table 3 with the corresponding row of Table 4. Under this mapping $wNAF$ produces a nonzero output iff $wNAF^*$ produces a nonzero output and $wNAF$ produces a nonzero carry-out iff $wNAF^*$ produces a nonzero carry out. Since the nonzero density of $wNAF$ is $\frac{1}{w+1}$ (see [10]), then the nonzero density of $wNAF^*$ must be $\frac{1}{w+1}$. •

Remark. In the appendix we illustrate how to compute the ECC scalar multiple (see Algorithm 5 in Appendix). In our algorithm, precomputations of the form $R_{frac} \cdot P$ will need to take place. This can be done by using a halving a point technique. For example, over binary elliptic curves the halving point algorithm [8] is very efficient, see [6,4]. However we could avoid the use of fractional symbols and use integers (see Example 2, row labeled with a). By doing so we benefit from the fact we have odd integers less than 2^{w-1} (in absolute value). Further this generalization of $wNAF^*$ will have the same nonzero density as $wNAF^*$. Thus this generalization has the same efficiency as $wNAF^*$ (in the appendix we construct the scalar multiple algorithm using this technique, see Algorithm 5). In general, $wNAF^*$ would require one more precomputation than $wNAF$ since $2P$ would need to be computed and stored, but again $2P$ would always have to be computed. Thus no extra precomputations are needed.

Table 5 illustrates the resource requirement of various signed digit recoding representations. This table was provided in [10], we have added the additional entry for the resource requirements of our algorithm $wNAF^*$. Only wMOF, left-to-right wNAF b and $wNAF^*$ are left-to-right w -ary recoding. However left-to-right wNAF b , has additional memory requirements. Most importantly $wNAF^*$ is much more straightforward to implement than wMOF.

Table 5. Table of various signed digit representations as provided in [10]

Scheme	Precomputations	density	additional memory
wNAF [12,2]	2^{w-2}	$\frac{1}{w+1}$	$O(n)$
Koyama [7]	$2^{w-1} - 1$	$\frac{1}{w+3/2}$	$O(n)$
NAF +SW [3]	$1/3(2^w + (-1)^{w+1})$	$\frac{1}{w+4/3-\nu(w)}$	$O(n)$
wMOF [10]	2^{w-2}	$\frac{1}{w+1}$	$O(w)$
left-to-right wNAF b [10]	2^{w-2}	$\frac{1}{w+1}$	$O(\frac{\log(w)}{w} \cdot n)$ when $w > 2$ $O(\log n)$ when $w=2$
$wNAF^*$	2^{w-2}	$\frac{1}{w+1}$	$O(w)$

Here $\nu(w) = \frac{(-1)^w}{3 \cdot 2^{w-2}}$.

5 Conclusion

We have examined an open problem concerning the creation of a left-to-right on-the-fly implementation of $wNAF$ by defining $wNAF^*$ and creating the left-to-right $wNAF^*$ recoding algorithm. Though the $wNAF^*$ recoding definition is not the same as the $wNAF$ definition, it closely mimics the definition. Further the left-to-right recoding algorithm for $wNAF^*$ closely mimics the right-to-left recoding algorithm of $wNAF$. Our work has provided the relationship between a right-to-left $wNAF$ calculation and the left-to-right w -ary non adjacent form calculation.

When applying the $wNAF^*$ representation to the key k in the computation of the ECC scalar multiple, it at first appears to require the use of halving of a point computation [8], which is very efficient. However it is possible to modify

our algorithm to create integer symbols, yet still reaping the same efficiency as $wNAF^*$. Our algorithm provides the optimal nonzero density (which implies less additions need to take place) while also maintaining the optimal minimal amount of precomputations (as illustrated in Table 5).

References

1. Arno, S., Wheeler, F.: Signed Digit Representations of Minimal Hamming Weight. *IEEE Transactions on Computers* 42(8), 1007–1009 (1993)
2. Blake, I., Seroussi, G., Smart, N.: *Elliptic Curve Cryptography*. Cambridge University Press, Cambridge (1999)
3. De Win, E., Mister, S., Preneel, B., Wiener, M.: On the performance of signature schemes based on elliptic curves. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 252–266. Springer, Heidelberg (1998)
4. Hankerson, D., Menezes, A., Vanstone, S.: *Guide to Elliptic Curve Cryptography*. Springer, New York (2004)
5. Joye, M., Yen, S.-M.: Optimal left-to-right binary signed-digit recoding. *IEEE Transactions on Computers* 49(7), 740–748 (2000)
6. King, B., Rubin, B.: Improvements to the halving a point algorithm. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 262–276. Springer, Heidelberg (2004)
7. Koyama, K., Tsuruoka, Y.: Speeding up Elliptic Cryptosystems by Using a Signed Binary Window Method. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 345–357. Springer, Heidelberg (1993)
8. Knudsen, E.W.: Elliptic Scalar Multiplication Using Point Halving. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 135–149. Springer, Heidelberg (1999)
9. Knuth, D.: *Art of Programming*, 3rd edn., NY (1998)
10. Okeya, K., Schmidt-Samoa, K., Spahn, C., Takagi, T.: Signed binary representations revisited. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Heidelberg (2004)
11. Reitwiesner, G.W.: Binary arithmetic. *Advances in Computers* 1, 231–308 (1960)
12. Solinas, J.: Efficient Arithmetic on Koblitz Curves. *Des. Codes Cryptography* 19(2/3), 195–249 (2000)

Appendix

We have modified the algorithm from Table 2 slightly to provide a more straightforward implementation of the algorithm, collapsing more than one case (i.e. more than one row of Table 2) into one logical statement.

We have modified the algorithm from Table 4 slightly to provide a more straightforward implementation of the algorithm, collapsing more than one case (i.e. more than one row of Table 2) into one logical statement, this is provided in Algorithm 5. In Algorithm 6, we have modified Algorithm 5, so that the symbols produced will be integers. We are using a technique illustrated in Example 2, row denoted by a and then incorporate this technique to compute the scalar multiple of an EC point.

Algorithm 4. Using NAF^* to compute the scalar multiple in a left-to-right fashion

```
1: INPUT:  $k = d_{n-1}, \dots, d_0$ , and base point  $P$  belonging to elliptic curve
2: OUTPUT:  $kP$ 
3: Compute  $2P$  and store it
4:  $c \leftarrow 0$ 
5:  $Q \leftarrow \mathcal{O}$  {  $\mathcal{O}$  is the point of infinity }
6:  $j \leftarrow n - 1$ 
7: while  $j \geq 0$  do
8:   if  $2 * d_j + c = 0$  then
9:      $Q \leftarrow 2Q$  {the next carry-in is the same as the previous carry-in}
10:  else
11:     $Q \leftarrow 2Q$ 
12:     $Q \leftarrow Q + (c + d_j + d_{j-1})P$ 
13:    if exactly one of  $c, d_j, d_{j-1}$  is nonzero then
14:       $c = 0$ 
15:    else
16:       $c = -2$ 
17:     $j \leftarrow j - 1$ 
    {Comment we now handle the case if  $\delta_{-1}$  is nonzero}
18:  if  $c \neq 0$  then
19:     $Q \leftarrow Q + (c/2)P$ 
20:  return  $Q$ 
```

Algorithm 5. Computing the EC scalar multiple using the $wNAF^*$ algorithm

```

1: INPUT:  $k = d_{n-1}, \dots, d_0$ , and base point  $P$  belonging to elliptic curve
2: OUTPUT:  $kP$ 
3: FORALL nonzero  $w$  bit integers with a leading one and trailing zero  $r =$ 
    $(1, x_1, \dots, x_{w-2}, 0)$ 
4:   Compute  $r_{frac}P$  and store in table
5: Compute  $2P$  and store in table
6:  $c \leftarrow 0$ 
7:  $Q \leftarrow \mathcal{O}$  {  $\mathcal{O}$  is the point of infinity }
8:  $j \leftarrow n - 1$ 
9:  $\rho \leftarrow -1$ 
10: while  $j \geq 0$  do
11:    $r = (d_j, d_{j-1}, \dots, d_{j-w+1})$ 
12:   if  $2 * d_j + c = 0$  then
13:      $Q \leftarrow 2Q, j \leftarrow j - 1$ 
14:   else
15:     if  $c = 0$  then
16:       if  $d_j \neq d_{j-w+1}$  then
17:          $t \leftarrow r_{frac}$ 
18:       else
19:          $t \leftarrow (r + 1)_{frac}$ 
20:          $c = -2$ 
21:     else
22:       if  $d_j = d_{j-w+1}$  then
23:          $t \leftarrow -(2^w - r)_{frac}$ 
24:          $c = 0$ 
25:     else
26:        $t \leftarrow -(2^w - (r + 1))_{frac}$ 
27:     if  $j \geq w$  then
28:        $Q \leftarrow 2^w Q$ 
29:        $Q \leftarrow Q + tP$  {use precomputed table to find  $tP$ }
30:        $j \leftarrow j - w$ 
31:     else
32:        $\rho \leftarrow j$ 
33:        $j \leftarrow j - w$ 
34:   if  $0 \leq \rho < w$  then
35:      $Q \leftarrow 2^\rho Q$ 
36:      $Q \leftarrow Q + tP$  {use precomputed table to find  $tP$ }
37:   else
38:     if  $j = -1$  and  $c = -2$  then
39:        $Q \leftarrow Q + -P$ 
40: return  $Q$ 

```

Algorithm 6. Modified version of scalar multiple calculation using $wNAF^*$ and integer constants

```

1: INPUT:  $k = d_{n-1}, \dots, d_0$ , and base point  $P$  belonging to elliptic curve
2: OUTPUT:  $kP$ 
3: FORALL nonzero  $w - 1$  (or less) bit odd integers  $r$ 
4:   Compute  $r \cdot P$  and store in table
5: Compute  $2P$  and store in table {This was actually done some time in line 2}
6:  $c \leftarrow 0$ 
7:  $Q \leftarrow \mathcal{O}$  {  $\mathcal{O}$  is the point of infinity }
8:  $j \leftarrow n - 1$ 
9:  $\rho \leftarrow -1$ 
10:  $\tau \leftarrow 0$ 
11: while  $j \geq 0$  do
12:    $Q \leftarrow 2^\tau Q$ 
13:    $r = (d_j, d_{j-1}, \dots, d_{j-w+1})$ 
14:   if  $2 * d_j + c = 0$  then
15:      $Q \leftarrow 2Q, j \leftarrow j - 1, \tau \leftarrow 0$ 
16:   else
17:     if  $c = 0$  then
18:       if  $d_j \neq d_{j-w+1}$  then
19:         Let  $\tau$  denote the least significant nonzero bit of  $r$  (label the LSB of a  $w$ 
           bit integer as the 0 bit)
20:          $t \leftarrow (r/2^\tau)$ 
21:       else
22:         Let  $\tau$  denote the least significant nonzero bit of  $r + 1$  (label the LSB of
           a  $w$  bit integer as the 0 bit)
23:          $t \leftarrow (r + 1)/2^\tau$ 
24:          $c = -2$ 
25:       else
26:         if  $d_j = d_{j-w+1}$  then
27:           Let  $\tau$  denote the least significant nonzero bit of  $2^w - r$  (label the LSB of
           a  $w$  bit integer as the 0 bit)
28:            $t \leftarrow -(2^w - r)/2^\tau$ 
29:            $c = 0$ 
30:         else
31:           Let  $\tau$  denote the least significant nonzero bit of  $2^w - (r + 1)$  (label the
           LSB of a  $w$  bit integer as the 0 bit)
32:            $t \leftarrow -(2^w - (r + 1))/2^\tau$ 
33:         if  $j \geq w$  then
34:            $Q \leftarrow 2^{w-\tau} Q$ 
35:            $Q \leftarrow Q + tP$  {use precomputed table to find  $tP$ }
36:            $j \leftarrow j - w$ 
37:         else
38:            $\rho \leftarrow j$ 
39:            $j \leftarrow j - w$ 
40:         if  $0 \leq \rho < w$  then
41:            $Q \leftarrow 2^{\rho-\tau} Q$ 
42:            $Q \leftarrow Q + tP$  {use precomputed table to find  $tP$ }
43:            $Q \leftarrow 2^\tau Q$ 
44:         else
45:           if  $j = -1$  and  $c = -2$  then
46:              $Q \leftarrow Q + -P$ 
47:         return  $Q$ 

```

A Very Compact “Perfectly Masked” S-Box for AES

D. Canright¹ and Lejla Batina²

¹ Applied Math., Naval Postgraduate School, Monterey CA 93943, USA,
dcanright@nps.edu

² K.U. Leuven ESAT/COSIC, Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium

Abstract. Implementations of the Advanced Encryption Standard (AES), including hardware applications with limited resources (e.g., smart cards), may be vulnerable to “side-channel attacks” such as differential power analysis. One countermeasure against such attacks is adding a random mask to the data; this randomizes the statistics of the calculation at the cost of computing “mask corrections.” The single nonlinear step in each AES round is the “S-box” (involving a Galois inversion), which incurs the majority of the cost for mask corrections. Oswald et al. [1] showed how the “tower field” representation allows maintaining an additive mask throughout the Galois inverse calculation. This work applies a similar masking strategy to the most compact (unmasked) S-box to date [2]. The result is the most compact masked S-box so far, with “perfect masking” (by the definition of Blömer [3]) giving suitable implementations immune to first-order differential side-channel attacks.

Keywords: AES, S-box, masking, DPA, composite Galois field.

1 Introduction

In 2001 the National Institute of Standards and Technology adopted the Rijndael algorithm [4] as the Advanced Encryption Standard (AES) [5], to provide a standard algorithm for secure encryption, intended not only for U.S. government documents, but also for electronic commerce. Since then, applications of AES have become widespread.

Many different implementations of AES have appeared, to satisfy the varying criteria of different applications. Some approaches seek to maximize throughput, e.g., [6,7,8,9]; others minimize power consumption, e.g., [10]; and yet others minimize circuitry, e.g., [11,12,13,14,15]. For the latter goal, Rijmen [16] suggested using subfield arithmetic in the crucial step of computing an inverse in the Galois Field of 256 elements. This idea was further extended by Satoh et al. [12], using sub-subfields (the “tower field” representation of Paar [17], also called the “composite-field” approach), along with other innovative optimizations, which resulted in the smallest AES circuit at that point. The S-box architecture of Satoh was refined somewhat by Canright [2], mainly through carefully chosen normal bases, resulting in the most compact S-box to date.

No attacks have yet been found on the AES algorithm itself that are more effective than exhaustive key search (“brute force”), although research continues, for example, on algebraic attacks [18,19]. But specific implementations of cryptography in software or hardware, e.g. in smart cards, may be vulnerable to “side-channel attacks” such as differential power analysis [20,21], that use statistical analysis of physical quantities such as power consumption, electromagnetic radiation, etc., to deduce information about the secret key.

One countermeasure against side-channel attacks is masking the data during calculation, through adding or multiplying by random values. All the steps in a round of AES are affine, except for the Galois field inversion substep of the S-box (*SubBytes*) step. For the other steps, calculation of the mask correction is linear, so an additive mask is most convenient. Some have suggested switching to a multiplicative mask for the Galois inverse step (e.g., [22]), but one inescapable weakness is that a zero data byte is unmasked by multiplication [23].

Applying the “tower field” representation, inversion in $GF(2^8)$ involves several multiplications and one inversion in the subfield $GF(2^4)$, which in turn involves multiplications and inversion in $GF(2^2)$. In the sub-subfield $GF(2^2)$, inversion is identical to squaring, and so is *linear* (over $GF(2)$). Oswald *et al.* [1] applied this idea to additive masking of the Galois inverse, and showed how to compute the mask corrections for the tower field approach. (Morioka and Akishita [24] apparently developed a similar masking scheme.) Many of the correction terms involve multiplication in subfields, and Oswald *et al.* showed how some of these multiplications can be eliminated through clever re-use of parts of the input mask for the output.

The present work incorporates this masking approach into the compact S-box of Canright [2], and also applies the optimization methods of [2] to the mask correction terms. At the level of operations in the subfield $GF(2^4)$, we simplify the approach somewhat, eliminating one multiplication and some additions, with further simplifications at lower levels. Even so, we show that our approach achieves the goal of “perfect masking,” in the terminology of Blömer *et al.* [3]: each intermediate result has a statistical distribution that is independent of the plaintext and key (assuming a source of uniformly distributed truly random masks). This level of security (a strengthened version of that in [25]) ensures that no first-order differential side-channel attacks can succeed, at least at the algorithmic level. (Higher-order attacks are possible, but take much greater effort.)

However, Mangard *et al.* [26] showed that first-order DPA attacks can succeed against a masked S-box using standard CMOS technology, even when the intermediate results at the algorithmic level are provably secure. The attack exploits glitches in the gate transition timings. In later work, Mangard *et al.* [27] showed that the information leakage is caused by specific XOR gates in masked multipliers, and can be eliminated if those XORs are made to satisfy timing constraints, either through delay elements or by enable signals. Also, rather than CMOS, other (more expensive) logic styles can be used to eliminate this potential problem. So masked S-boxes can be made secure from first-order DPA. At any rate,

the current work only considers the intermediate results at the algorithmic level, and shows that they are provably secure, which is the best that can be expected.

We apply the same optimizations as in the unmasked S-box of [2], including efficient normal bases, elimination of common sub-expressions, and logic-gate substitutions, to the masked S-box calculation here, including mask correction terms. The result is, as far as we know, the most compact *masked* S-box to date (for the 0.13- μ CMOS standard cell library considered). But the cost for this security is that the S-box size is almost three times that of the unmasked S-box. Also, since the security requires certain calculations to proceed in sequence, the speed will be reduced. For applications with sufficient resources to unroll the round loop, we show how re-using masks *between rounds* can save a significant amount of the mask correction calculations; then the masked S-box is roughly twice the size of the unmasked. Our compact masked S-box design could be useful for securing some hardware AES applications, especially those with limited resources, against first-order differential attacks.

2 Algebraic Description

The AES algorithm has been described thoroughly and frequently elsewhere [5]; here we give the barest outline before concentrating on the S-box. It is a symmetric block cipher (16 bytes, though the original Rijndael cipher supports other block sizes [4]) consisting of several rounds (10, 12, or 14, for a key size 16, 24, or 32 bytes, respectively). Each round involves the four steps called *SubBytes* (byte substitution, or S-box), *ShiftRows*, *MixColumns*, and *AddRoundKey* (the last round skips *MixColumns*, and there is a Round 0 consisting solely of *AddRoundKey*). The latter three steps are linear with respect to the data block, and provide “diffusion.” *SubBytes* is the nonlinear step that provides “confusion.”

The S-box, applied to each byte, consists of two substeps: (i) considering the byte an element of the Galois field $GF(2^8)$, find its inverse in that field (except a zero byte, which has no inverse, remains unchanged); (ii) considering the resulting byte a vector of bits in $(GF(2))^8$, multiply by a given bit matrix and add a given constant vector, i.e., an affine transformation.

In the particular Galois field of AES, a byte represents a polynomial where the bits are coefficients of corresponding powers of x , and multiplication is modulo the irreducible polynomial $q(x) = x^8 + x^4 + x^3 + x + 1$. Equivalently, one could consider a root, say θ , of this polynomial, so $q(\theta) = 0$ in this field; then the bits of a byte would correspond to coefficients of powers of θ , e.g., $2 = \theta$, $3 = \theta + 1$, $4 = \theta^2$, etc. Thus the bits form a vector with respect to what is called a polynomial basis. But there are computational advantages to considering a different (though isomorphic) representation of $GF(2^8)$. Instead of a vector of dimension eight over $GF(2)$, we consider a byte as a vector of dimension two over $GF(2^4)$, where each 4-bit element is in turn a vector of dimension two over $GF(2^2)$, and finally each 2-bit element is a vector of dimension two over $GF(2)$. This has been called a composite field, or “tower field” representation [17]. In this way, the 8-bit inverse calculation comprises several 4-bit operations, each consisting of various simple

2-bit calculations. For each of these subfields, it has been shown [2] that a normal basis (consisting of a conjugate pair) is more efficient than a polynomial basis for the required inverse calculation; for each particular basis used, the trace (sum of the conjugate pair) is 1, and the norm (product of the conjugate pair) is a nonzero element of the subfield [2].

Converting between the standard AES representation and the composite field representation amounts to a change of basis, accomplished by multiplying the bit vector by a bit matrix. In converting back, this bit matrix can be combined with that of the affine transformation substep [12]. With regard to an additive mask, these matrix multiplications are simple linear calculations for the mask correction terms. Below we detail the mask corrections required for the nonlinear inverse calculation.

2.1 Inversion without Masking

Here we employ the following convention: upper-case bold symbols represent elements of the main field (e.g. $\mathbf{A} \in GF(2^8)$); upper-case italic symbols are for elements of the subfield (e.g. $A \in GF(2^4)$); lower-case bold is used for the sub-subfield (e.g. $\mathbf{a} \in GF(2^2)$); and lower-case italic is for single bits (e.g. $a \in GF(2)$).

Without masking, inversion in $GF(2^8)/GF(2^4)$ (indicating the representation of $GF(2^8)$ as vectors over $GF(2^4)$) using a normal basis $[\mathbf{Y}^{16}, \mathbf{Y}]$, where \mathbf{Y} and \mathbf{Y}^{16} are the roots of $\mathbf{X}^2 + \mathbf{X} + N$ and $N \in GF(2^4)$ is the norm (product: $N = \mathbf{Y}^{17}$), is given by [2]:

$$\begin{aligned} \mathbf{A} &= A_1 \mathbf{Y}^{16} + A_0 \mathbf{Y} \quad (\text{given}) \quad , & (1) \\ B &= N \otimes (A_1 \oplus A_0)^2 \oplus A_1 \otimes A_0 \quad , & (2) \\ \mathbf{A}^{-1} &= (A_0 \otimes B^{-1}) \mathbf{Y}^{16} + (A_1 \otimes B^{-1}) \mathbf{Y} \quad (\text{result}) \quad . & (3) \end{aligned}$$

(Note that \otimes and \oplus denote multiplication and addition calculations in a Galois field, while $A_1 \mathbf{Y}^{16} + A_0 \mathbf{Y}$ is just the algebraic expression for the vector $[A_1, A_0]$ in the normal basis.) This requires inversion, multiplication, and the combined “square-scale” operation ($N \otimes X^2$) in the subfield $GF(2^4)$. Similarly, the inversion in $GF(2^4)/GF(2^2)$ using a normal basis $[Z^4, Z]$, where Z and Z^4 are the roots of $X^2 + X + \mathbf{n}$ and $\mathbf{n} \in GF(2^2)$ is the norm ($\mathbf{n} = Z^5$), is given by:

$$\begin{aligned} B &= \mathbf{b}_1 Z^4 + \mathbf{b}_0 Z \quad (\text{given}) \quad , & (4) \\ \mathbf{c} &= \mathbf{n} \otimes (\mathbf{b}_1 \oplus \mathbf{b}_0)^2 \oplus \mathbf{b}_1 \otimes \mathbf{b}_0 \quad , & (5) \\ B^{-1} &= (\mathbf{b}_0 \otimes \mathbf{c}^{-1}) Z^4 + (\mathbf{b}_1 \otimes \mathbf{c}^{-1}) Z \quad (\text{result}) \quad . & (6) \end{aligned}$$

But in the sub-subfield $GF(2^2)$, inversion is the same as squaring, equivalent to a bit swap:

$$\begin{aligned} \mathbf{c} &= c_1 \mathbf{w}^2 + c_0 \mathbf{w} \quad (\text{given}) \quad , & (7) \\ \mathbf{c}^{-1} &= c_0 \mathbf{w}^2 + c_1 \mathbf{w} \quad (\text{result}) \quad , & (8) \end{aligned}$$

where \mathbf{w} and \mathbf{w}^2 are the roots of $\mathbf{x}^2 + \mathbf{x} + 1$. (While this algebraic description uses the Galois inverse, the case of a zero element in any of these fields is correctly handled: the zero element is returned in lieu of an inverse.)

2.2 Masked Inversion

Now introduce additive masking. By adding a random mask, such that the statistical distribution of masks is uniform over the field, now our operands appear random as well, uncorrelated to either plaintext or key. Hence the statistical data available through side channels appears as noise, independent of the chosen sets of plaintexts, and the key is protected against first-order differential attacks. The cost is the computation of mask correction terms that are combined with the given masked input to correctly mask the output. Here we outline the algebraic steps involved; we later show that this masking scheme is secure in [2.5](#).

We use the insight of Oswald *et al.* that in the sub-subfield $GF(2^2)$ inversion (squaring) is additive, so for data \mathbf{a} and mask \mathbf{m} , then

$$(\mathbf{a} \oplus \mathbf{m})^{-1} = (\mathbf{a} \oplus \mathbf{m})^2 = \mathbf{a}^2 \oplus \mathbf{m}^2 = \mathbf{a}^{-1} \oplus \mathbf{m}^{-1} , \quad (9)$$

and finding the mask correction \mathbf{m}^2 is trivial. Hence the tower-field approach eliminates the need to remove the additive mask (or change it to a multiplicative one) before inversion.

In the larger fields, here is how the mask corrections can be calculated. We indicate the masked version of the input byte \mathbf{A} with a tilde: $\tilde{\mathbf{A}}$, and similarly for the other masked quantities. So the input to the masked $GF(2^8)$ inverter is the data byte \mathbf{A} already masked by the (known) mask \mathbf{M}

$$\tilde{\mathbf{A}} = (\mathbf{A} \oplus \mathbf{M}) = \tilde{A}_1 \mathbf{Y}^{16} + \tilde{A}_0 \mathbf{Y} , \quad (10)$$

$$\mathbf{M} = M_1 \mathbf{Y}^{16} + M_0 \mathbf{Y} . \quad (11)$$

Let

$$\tilde{B} = N \otimes (\tilde{A}_1 \oplus \tilde{A}_0)^2 \oplus \tilde{A}_1 \otimes \tilde{A}_0 \oplus \tilde{A}_1 \otimes M_0 \oplus \tilde{A}_0 \otimes M_1 \oplus M_1 \otimes M_0 , \quad (12)$$

$$M_2 = N \otimes (M_1 \oplus M_0)^2 , \quad (13)$$

with the result \tilde{B} being B above, masked by M_2 , which is uniformly distributed. Here the terms must be added in sequence to keep each intermediate result uniformly distributed, as discussed below in [2.5](#).

For the subfield inversion, say $\tilde{B} = \tilde{\mathbf{b}}_1 Z^4 + \tilde{\mathbf{b}}_0 Z$ and $M_2 = \mathbf{m}_1 Z^4 + \mathbf{m}_0 Z$, and let

$$\tilde{\mathbf{c}} = \mathbf{n} \otimes (\tilde{\mathbf{b}}_1 \oplus \tilde{\mathbf{b}}_0)^2 \oplus \tilde{\mathbf{b}}_1 \otimes \tilde{\mathbf{b}}_0 \oplus \tilde{\mathbf{b}}_1 \otimes \mathbf{m}_0 \oplus \tilde{\mathbf{b}}_0 \otimes \mathbf{m}_1 \oplus \mathbf{m}_1 \otimes \mathbf{m}_0 , \quad (14)$$

so $\tilde{\mathbf{c}}$ is \mathbf{c} above, masked by $\mathbf{n} \otimes (\mathbf{m}_1 \oplus \mathbf{m}_0)^2$ (which need not be computed, only its square \mathbf{m}_2 below).

In the sub-subfield, say $\tilde{\mathbf{c}} = \tilde{c}_1 \mathbf{w}^2 + \tilde{c}_0 \mathbf{w}$, and let

$$\tilde{\mathbf{c}}^{-1} = \tilde{c}_0 \mathbf{w}^2 + \tilde{c}_1 \mathbf{w} \quad (\text{bit swap}) , \quad (15)$$

$$\mathbf{m}_2 = \mathbf{n}^2 \otimes (\mathbf{m}_1 \oplus \mathbf{m}_0) , \quad (16)$$

so $\tilde{\mathbf{c}}^{-1}$ is \mathbf{c}^{-1} above masked by another uniform mask \mathbf{m}_2 .

The next steps would involve only multiplications, but directly adding *any* of the correction terms would reveal a distribution that depends on the data. Hence (as in Oswald *et al.* [11]) we need to introduce another additive mask. This mask could be new, or could be re-used bits from the original mask \mathbf{M} . In either case, this mask must be added *first*, then the other mask correction terms added individually to the sum, to maintain the uniform distribution for intermediate results.

Say now we introduce a new temporary 4-bit mask $T = \mathbf{t}_1 Z^4 + \mathbf{t}_0 Z$, and let

$$\tilde{\mathbf{b}}_1^{-1} = \mathbf{t}_1 \oplus \tilde{\mathbf{b}}_0 \otimes \tilde{\mathbf{c}}^{-1} \oplus \tilde{\mathbf{b}}_0 \otimes \mathbf{m}_2 \oplus \mathbf{m}_0 \otimes \tilde{\mathbf{c}}^{-1} \oplus \mathbf{m}_0 \otimes \mathbf{m}_2, \quad (17)$$

$$\tilde{\mathbf{b}}_0^{-1} = \mathbf{t}_0 \oplus \tilde{\mathbf{b}}_1 \otimes \tilde{\mathbf{c}}^{-1} \oplus \tilde{\mathbf{b}}_1 \otimes \mathbf{m}_2 \oplus \mathbf{m}_1 \otimes \tilde{\mathbf{c}}^{-1} \oplus \mathbf{m}_1 \otimes \mathbf{m}_2, \quad (18)$$

so that the result $\tilde{B}^{-1} = \tilde{\mathbf{b}}_1^{-1} Z^4 + \tilde{\mathbf{b}}_0^{-1} Z$ is B^{-1} above, masked by T (but is *not* the inverse of \tilde{B}).

Similarly, introduce a new 8-bit mask $\mathbf{S} = S_1 \mathbf{Y}^{16} + S_0 \mathbf{Y}$ for the output, and let

$$\tilde{A}_1^{-1} = S_1 \oplus \tilde{A}_0 \otimes \tilde{B}^{-1} \oplus \tilde{A}_0 \otimes T \oplus M_0 \otimes \tilde{B}^{-1} \oplus M_0 \otimes T, \quad (19)$$

$$\tilde{A}_0^{-1} = S_0 \oplus \tilde{A}_1 \otimes \tilde{B}^{-1} \oplus \tilde{A}_1 \otimes T \oplus M_1 \otimes \tilde{B}^{-1} \oplus M_1 \otimes T, \quad (20)$$

so that the result $\tilde{\mathbf{A}}^{-1} = \tilde{A}_1^{-1} \mathbf{Y}^{16} + \tilde{A}_0^{-1} \mathbf{Y}$ is the answer \mathbf{A}^{-1} above, masked by the output mask \mathbf{S} :

$$\tilde{\mathbf{A}}^{-1} = \mathbf{A}^{-1} \oplus \mathbf{S}. \quad (21)$$

2.3 Re-using Masks

Oswald *et al.* [11] showed that through using parts of the input mask for the intermediate results and the output, then several operations can be eliminated, notably multiplications (for the cost of a few additions). We will follow the same strategy below. (While re-using masks could make the implementation more vulnerable to higher-order differential side-channel analysis, it remains secure against first-order attacks.)

The first place where re-using masks helps is in the masked intermediate result $\tilde{\mathbf{c}}^{-1}$, where for one subsequent calculation the mask \mathbf{m}_1 would be helpful but for another the preferred mask would be \mathbf{m}_0 , so we follow [11] and switch masks. Then starting at (15) above we modify the calculation as follows:

$$\tilde{\mathbf{c}}^{-1} = [\tilde{c}_0 \mathbf{w}^2 + \tilde{c}_1 \mathbf{w}] \oplus (\mathbf{m}_1 \oplus \mathbf{m}_2), \quad (22)$$

$$\tilde{\mathbf{b}}_1^{-1} = \mathbf{m}_{11} \oplus \tilde{\mathbf{b}}_0 \otimes \tilde{\mathbf{c}}^{-1} \oplus \underline{\tilde{\mathbf{b}}_0 \otimes \mathbf{m}_1} \oplus \mathbf{m}_0 \otimes \tilde{\mathbf{c}}^{-1} \oplus \underline{\mathbf{m}_0 \otimes \mathbf{m}_1}, \quad (23)$$

$$\tilde{\mathbf{c}}_2^{-1} = \tilde{\mathbf{c}}^{-1} \oplus (\mathbf{m}_0 \oplus \mathbf{m}_1), \quad (24)$$

$$\tilde{\mathbf{b}}_0^{-1} = \mathbf{m}_{10} \oplus \tilde{\mathbf{b}}_1 \otimes \tilde{\mathbf{c}}_2^{-1} \oplus \underline{\tilde{\mathbf{b}}_1 \otimes \mathbf{m}_0} \oplus \mathbf{m}_1 \otimes \tilde{\mathbf{c}}_2^{-1} \oplus \underline{\mathbf{m}_1 \otimes \mathbf{m}_0}, \quad (25)$$

where the underlined products had already been computed previously and may be re-used. (Parens indicate the order of evaluation necessary to avoid unmasking operands. In the actual optimized code, the details of the specific bits added are different.) The result $\tilde{B}^{-1} = \tilde{\mathbf{b}}_1^{-1} Z^4 + \tilde{\mathbf{b}}_0^{-1} Z$ is still B^{-1} above, but now masked

by $M_1 = \mathbf{m}_{11} Z^4 + \mathbf{m}_{10} Z$, the upper half of the input mask. Following the same approach of switching masks at the next level gives

$$\tilde{A}_1^{-1} = S_1 \oplus \tilde{A}_0 \otimes \tilde{B}^{-1} \oplus \underline{\tilde{A}_0 \otimes M_1} \oplus M_0 \otimes \tilde{B}^{-1} \oplus \underline{M_0 \otimes M_1} , \quad (26)$$

$$\tilde{B}_2^{-1} = \tilde{B}^{-1} \oplus (M_0 \oplus M_1) , \quad (27)$$

$$\tilde{A}_0^{-1} = S_0 \oplus \tilde{A}_1 \otimes \tilde{B}_2^{-1} \oplus \underline{\tilde{A}_1 \otimes M_0} \oplus M_1 \otimes \tilde{B}_2^{-1} \oplus \underline{M_1 \otimes M_0} , \quad (28)$$

again allowing the underlined products to be re-used, and with the output $\tilde{\mathbf{A}}^{-1} = \tilde{A}_1^{-1} \mathbf{Y}^{16} + \tilde{A}_0^{-1} \mathbf{Y}$ having the output mask \mathbf{S} (which could be the original input mask \mathbf{M} , or not):

$$\tilde{\mathbf{A}}^{-1} = \mathbf{A}^{-1} \oplus \mathbf{S} . \quad (29)$$

2.4 Re-using Masks Between Rounds

Many of the mask correction terms used in the masked inversion above involve *only* the input mask, independent of the masked data. This is also true of *all* the mask correction term calculations in the other steps of each round of encryption, as those other steps are all linear (with respect to the additive mask). Then, if the original 128-bit mask for a block of data were to be re-used for every round, all those data-independent correction terms would be the same for each round. For implementations where the round loop is “unrolled,” with S-boxes for each round, these terms would only need computing once, then could be passed along to all the other rounds. This would save the re-computation of all those mask terms, eliminating the associated circuitry, at the modest cost of the “wiring” required to pass along the correction terms. Of course, one would use a new random mask with each new block of data in Round 0, to ensure that over time the distribution of masks remains uniform.

More precisely, one way to do this starts by picking a random 128-bit mask that will be used as the *output* mask (whose bytes correspond to \mathbf{S} above) from the inversion step. Then after each byte undergoes the basis change (from the tower field form) combined with affine transformation part of the S-box (excluding the additive constant), the *ShiftRows* step is applied to the whole mask; the result is the output mask after the last round of encryption (which lacks the *MixColumns* step). Then *MixColumns* is applied to that, giving the *input* mask to be added to the initial data before Round 0. Applying byte-wise the basis change (to the tower field form) gives the input mask (corresponding to \mathbf{M} above) for the inversion step. From this can be computed such terms as $M_1 \otimes M_0$, M_2 , $\mathbf{m}_1 \otimes \mathbf{m}_0$, and \mathbf{m}_2 above, to be re-used each round. Then the only correction terms that would need computing in each round are the data-dependent terms (e.g. $\tilde{A}_1 \otimes M_0$ above) of the inversion step.

But this only makes sense if the application has enough room to unroll (at least partly) the round loop. (While unrolling the rounds does not improve latency, this saving of correction terms may make unrolling preferable to simply duplicating more encryptors for increased throughput.) In cases where compactness is paramount the same few S-boxes would be employed for each round; using pre-computed correction terms from round to round would then require extra registers – a cost rather than a saving.

2.5 Security of Masked Operands

Here we show that the masked inversion operation outlined above is secure, by which we mean, assuming a source of truly random uniformly distributed masks, then the distribution of each intermediate result is independent of both the plaintext data and the key. This gives “perfect masking” in the terminology of [3], and hence protection from first-order differential side-channel attacks. We start with Lemmas 1 and 2 of [3] (paraphrased) and, to be thorough, add two more lemmas to cover all the operations in inversion, which are then examined in detail.

Lemma 1. *Given x uniformly distributed over a finite field \mathbb{F} , and any $y \in \mathbb{F}$ independent of x , then $z = x \oplus y$ is also uniformly distributed and independent of y . [3]*

Lemma 2. *Given x and y independent and both uniformly distributed over a finite field $GF(p^n)$, then $z = x \otimes y$ is distributed according to*

$$Pr(z = i) = \begin{cases} (2p^n - 1)/p^{2n}, & i = 0 \\ (p^n - 1)/p^{2n}, & i \neq 0 \end{cases}$$

here called the random product distribution. [3]

Lemma 3. *Given x uniformly distributed over a finite set \mathbb{F} , and a one-to-one mapping $f : \mathbb{F} \rightarrow \mathbb{F}$, then $y = f(x)$ is also uniformly distributed.*

Proof: For a finite set, any one-to-one mapping is a bijection, i.e., just a permutation of the elements, so a uniform distribution is unchanged. (Note in particular that any isomorphism of a finite field is a bijection.)

Lemma 4. *Given $\mathbf{x} = [x_1, x_2, \dots, x_{2n}]$ uniformly distributed over the set \mathbb{F}^{2n} of ordered $2n$ -tuples from a finite set \mathbb{F} , then the two halves $\mathbf{y}_1 = [x_1, x_2, \dots, x_n]$ and $\mathbf{y}_2 = [x_{n+1}, x_{n+2}, \dots, x_{2n}]$ of \mathbf{x} are independent and uniformly distributed over \mathbb{F}^n .*

Proof: Since \mathbf{x} is uniform if and only if each x_i is independently uniform over \mathbb{F} , then \mathbf{y}_1 and \mathbf{y}_2 are independent and uniform. (So given a uniform mask of n bits then any sub-mask is also uniform.)

Consider the operations in the masked inversion above. Initially adding a uniform mask to the plaintext data gives a uniform result, as is $\tilde{\mathbf{A}}$ in (10) above, with independent uniform halves \tilde{A}_1, \tilde{A}_0 . Then $\tilde{A}_1 \oplus \tilde{A}_0$ is uniform, so is its square (squaring is an isomorphism in a field of characteristic 2), and so is that square scaled by the norm N , since multiplication by a nonzero constant is a one-to-one mapping. Each of the other four products in (12) for \tilde{B} has the random product distribution.

Now the order of adding these pieces together is crucial, because adding any pair of those four products would give a distribution that depends on the data. So the uniform $N \otimes (\tilde{A}_1 \oplus \tilde{A}_0)^2$ must be added to the first product *before* any other

products are added; by successively adding each other product to the previous uniform sum, then each resulting sum is uniform, including \tilde{B} . Also, its mask $M_2 = N \otimes (M_1 \oplus M_0)^2$ is uniform.

Similar reasoning applies to \tilde{c} in (14). Then its square in (22) is also uniform, with the uniform mask $\mathbf{m}_2 = \mathbf{n}^2 \otimes (\mathbf{m}_1 \oplus \mathbf{m}_0)$; hence the order of summation for \tilde{c}^{-1} in (22) is important to avoid unmasking by $(\tilde{c})^2 \oplus \mathbf{m}_2$. Again, in (23) for \tilde{b}_1^{-1} , each of the four products has the random product distribution, and the *first* sum must be with the uniform mask \mathbf{m}_{11} , then each other product added successively, so each result is uniform. And again, in switching masks for \tilde{c}_2^{-1} in (24) the order of summation for is important to prevent unmasking by $\tilde{c}^{-1} \oplus \mathbf{m}_1$. All the remaining steps are similar to those already discussed.

Therefore, the result of every calculation is either uniformly distributed or has the random product distribution (provided the summations are performed in the correct order), independent of the data and key: “perfect masking.” This protects suitable implementations from attacks by first-order differential side-channel analysis. (Resistance against higher-order attacks would be improved by avoiding re-use of masks within the S-box, as well as between rounds.)

3 Optimizations and Results

The algebraic description above shows the most efficient masked inversion at that algebraic, hierarchical level. At the bit level, further optimization is possible due to certain bit combinations being useful in more than one place. And at the logic gate level, certain substitutions of types of logic gates give further savings in circuit size (for the standard cell library considered).

Here we briefly describe the main optimizations, which are essentially the same as in [2], but applied also to mask correction calculations. Each Galois multiplier involves bit sums, and since all factors are shared between two multipliers (using normal bases), these bit sums can be re-used; some of them are also useful as parts of the square-scale operation. And bit sums required for mask switching in (27) are useful in subfield operations also. Gate-level optimizations (minimizing the size for the 0.13- μ CMOS standard cell library [28] considered) include replacing AND by NAND, and where possible, replacing a NAND and some XORs by a single NOR. (We only consider two-input logic gates; using gates with more inputs may allow some improvement.)

But the S-box involves more than inversion in the tower-field representation. The affine transformation and conversion to/from the tower-field form are also required. (While one approach is to use the tower-field form for the entire encryption round [11], we use it only for the inversion, so that the *MixColumns* step remains simple.) For decryption, the inverse affine transformation is required. The conversion between the standard form and the tower-field form can be done through multiplying the byte by a bit matrix, and one of those two matrices can be combined with the bit matrix multiply of the affine transformation (or inverse). Satoh *et al.* [12] showed how, when both encryption and decryption are required, an architecture sharing a single Galois inverter between an S-box and

an inverse S-box allows some further optimization in the four matrices involved. We use the particular normal bases of [2] and the corresponding optimally factored matrices to minimize this computation.

We wrote a complete implementation, with all optimizations, of the masked S-box and inverse S-box using the merged architecture of [12], with a shared Galois inverter. The complete implementation, written in Verilog, is given in [29]. Here both the input mask and the output mask are parameters, along with the masked data byte. The code has been compiled and run on an FPGA, and shown to give correct results for every combination of encryption/decryption, data byte, input mask, and output mask (33,554,432 combinations).

Tables 1 and 2 give the resulting numbers of logic gates separately for the masked Galois inverter and the basis change (bit matrices). Results are shown by number and type of specific logic operations, and also by total “gates,” where the number refers to the equivalent number of NAND gates (rounded to whole numbers), using our standard cell library. We use the equivalencies 1 XOR/XNOR = $\frac{7}{4}$ NAND gates, 1 NOR = 1 NAND gate, 1 NOT = $\frac{3}{4}$ NAND gate, and 1 MUX2I1 = $\frac{7}{4}$ NAND gates [28].

Table 1. Inverter Size. Here we compare the masked inverter with the unmasked version, where total gates is in NAND equivalents.

Inverter	gate counts	total gates
masked	217 XOR, 94 NAND, 6 NOR	480
unmasked	56 XOR, 34 NAND, 6 NOR	138

Table 2. Basis Change Sizes. Here we compare gates needed in the basis change bit matrices (including the affine transformation but excluding the Galois inverter) for a merged S-box & inverse, S-box alone, and inverse S-box alone, using different input and output masks, same mask for both, or no mask. Both individual gate counts and NAND equivalents are given.

Basis Change	merged	S-box	(S-box) ⁻¹
2 masks	78 XOR, 4 NOT, 32 MUX = 196	49 XOR = 86	50 XOR = 88
1 mask	58 XOR, 3 NOT, 32 MUX = 160	44 XOR = 77	45 XOR = 79
unmasked	38 XOR, 2 NOT, 16 MUX = 96	24 XOR = 42	25 XOR = 44

Note that the additional resources needed to use different masks on input and output are significant for the merged architecture, but not for dedicated encryption (or decryption) only. For protection against first-order differential attacks, there is no reason not to use the input mask for the output as well. In this case, the size for the merged architecture where encryption and decryption share an inverter is 640 NAND equivalents, nearly three times the size of the unmasked version (234 gates). (For encryption only, not merged, the S-box with a single mask for both input and output is 557 NAND equivalents, compared with 180 for unmasked; again masked is three times larger.)

However, if the current approach were used in an application where the loop of rounds was “unrolled” (requiring enough room for at least 160 S-boxes for complete unrolling), the masks could be re-used from round to round, as discussed above in [2.4](#). This would require passing along the extra bits of pre-computed corrections between rounds. For one S-box, the total number of mask-term bits would be 43, as compared to 8 bits for an input mask alone (to be used as output mask also, or 16 bits for two different masks). These extra wires would replace 33 XORs and 12 NANDs in the inverter, and *all* of the mask basis change calculation (so the basis change would be as if unmasked). The gains from this re-use between rounds is shown in [Table 3](#); then a masked merged S-box is 506 NAND equivalents, rather than the 640 above. In addition to this saving per S-box (after the first round), the *MixColumns* operation on the mask block would also be eliminated (again, after the first round).

Table 3. Gains from re-using masks *between* rounds, for the complete S-box, in NAND equivalents. (This re-use requires unrolling the round loop with many copies of the S-box.)

masking	merged	S-box	(S-box) ⁻¹
1 mask	640	557	559
re-use	506	452	454

Table 4. High-level comparison of masking schemes: $GF(2^4)$ operations, from [Table 1](#) of [\[1\]](#) with a new row for the current work, and a new column for the Square-Scale optimized combination [addition and inverse operations not shown].

method	Mult	MultConst	Square	Square-Scale
S-Akkar	18	6	4	0
S-Blömer	12	1	2	0
MS-IAIK	9	2	2	0
this work	8	0	0	2

Direct comparison with Oswald *et al.* [\[1\]](#) is difficult at the level of optimization employed here. Their terms of comparison are operations in $GF(2^4)$ and their [Table 1](#) is reproduced here as [Table 4](#), augmented to compare with the present work. They compare their approach (MS-IAIK) to their implementations of two previous methods, and do not include addition and inversion operations, presumably because addition is relatively small (4 XORs) and one subfield inversion is assumed. The column for multipliers in $GF(2^4)$ is the most significant (since each $GF(2^4)$ multiplier includes 3 multipliers and 4 additions in $GF(2^2)$) and our approach saves one multiplier at this level. It is not clear how they implemented squaring, and multiplication by a constant, called “scaling” here; in our approach with normal bases, squaring is always followed with scaling by the norm, so we treat square-scale as a single operation, which we have optimized down to only 3 XORs, less than one of the 4-bit additions that are not counted in [\[1\]](#).

Some rough comparison is possible in the algebraic description at the $GF(2^4)$ level: our (12) and (13) are approximately comparable to their (19), while our (26), (28) are like their (15), (17) respectively. In particular, their (19) (which includes 4 squares, though only 2 appear in [1, Table 1]) shows more terms than our combined (12,13). So we save at least some $GF(2^4)$ additions, and again at the $GF(2^2)$ level. But while detailed comparison is difficult in the lack of specifics, we are confident that ours is the smallest masked S-box to date, because it uses the same optimizations as the smallest unmasked S-box to date.

4 Conclusion

Side-channel attacks can be an major concern for certain applications of AES, including hardware applications with limited resources, such as smart cards. Adding random masks can be an effective countermeasure, though at the cost of computing mask corrections in the S-box (the rest of each round being linear). Here we show how to compute the Galois inverse (the nonlinear part of the S-box) with “perfect masking,” in that the distributions of *all* the masked operands are independent of the chosen plaintext and key; hence suitable implementations employing this method are secure against first-order differential side-channel attacks. (Though, as discussed in the Introduction, CMOS implementations might be vulnerable to DPA attacks, due to glitches [26], unless specific timing constraints are met [27].) While our approach is similar to [1], we have reduced the number of operations at every level. We have optimized this masked S-box for minimal chip area, giving the smallest masked S-box of which we are aware.

The overhead for masking nearly triples the size of the S-box, from 234 gates (NAND equivalents) to 640 gates for the merged version. In applications with sufficient resources to unroll the round loop (where the compactness of our S-box allows more copies for a given area), this overhead may be reduced through re-using the block mask between rounds. Then (after the first round) each S-box would require only 506 gates, a little over twice the size of the unmasked version, and the mask correction for the rest of each round would also be eliminated.

Acknowledgements

We would like to thank the reviewers for several helpful suggestions.

References

1. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A side-channel analysis resistant description of the AES S-box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
2. Canright, D.: A very compact S-box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
3. Blömer, J., Guajardo, J., Krummel, V.: Provably secure masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)

4. Daemen, J., Rijmen, V.: The Design of Rijndael, AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
5. NIST: Specification for the ADVANCED ENCRYPTION STANDARD (AES). Technical Report FIPS PUB 197, National Institute of Standards and Technology (NIST) (November 2001)
6. Morioka, S., Satoh, A.: A 10 Gbps full-AES crypto design with a twisted-BDD S-box architecture. In: IEEE International Conference on Computer Design, pp. 98–103 (2002)
7. Weaver, N., Wawrzynek, J.: High performance, compact AES implementations in Xilinx FPGAs (2002), http://www.cs.berkeley.edu/~nweaver/papers/AES_in_FPGAs.pdf
8. Jarvinen, K.U., Tommiska, M.T., Skytta, J.O.: A fully pipelined memoryless 17.8 Gbps AES128 encryptor. In: FPGA 2003, ACM, New York (2003)
9. Hodjat, A., Hwang, D., Lai, B.-C., Tiri, K., Verbauwhede, I.: A 3.84 Gbits/s AES crypto coprocessor with modes of operation in a 0.18- μ m CMOS technology. In: ACM Great Lakes Symposium on VLSI, pp. 60–63 (2005)
10. Morioka, S., Satoh, A.: An optimized S-box circuit architecture for low power AES design. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 172–186. Springer, Heidelberg (2003)
11. Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R., Rohatgi, P.: Efficient Rijndael encryption implementation with composite field arithmetic. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 171–184. Springer, Heidelberg (2001)
12. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with s-box optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
13. Wolkerstorfer, J., Oswald, E., Lamberger, M.: An ASIC implementation of the AES S-boxes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 67–78. Springer, Heidelberg (2002)
14. Chodowicz, P., Gaj, K.: Very compact FPGA implementation of the AES algorithm. In: D. Walter, C., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 319–333. Springer, Heidelberg (2003)
15. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES implementation on a grain of sand. In: IEE Proceedings on Information Security, vol. 152, pp. 13–20 (2005)
16. Rijmen, V.: Efficient implementation of the Rijndael S-box (2001), <http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf>
17. Paar, C.: Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. PhD thesis, Institute for Experimental Mathematics, University of Essen, Germany (1994)
18. Courtois, N., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
19. Murphy, S., Robshaw, M.J.B.: Essential algebraic structure within the AES. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 1–16. Springer, Heidelberg (2002)
20. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
21. Akkar, M.L., Bévan, R., Dischamps, P., Moyart, D.: Power Analysis, What is Now Possible... In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 489–502. Springer, Heidelberg (2000)

22. Akkar, M.L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
23. Golić, J., Tymen, C.: Multiplicative masking and power analysis of AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 198–212. Springer, Heidelberg (2003)
24. Morioka, S., Akishita, T.: DPA attack to AES S-box circuits over composite fields. *Joho Shori Gakkai Shinpojiumu Ronbunshu* 2004(11) 9C–2 (2004) (in Japanese)
25. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
26. Mangard, S., Pramstaller, N., Oswald, E.: Successfully attacking masked AES hardware implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
27. Mangard, S., Schramm, K.: Pinpointing the side-channel leakage of masked AES hardware implementations. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 76–90. Springer, Heidelberg (2006)
28. Satoh, A.: personal communication (July 2004)
29. Canright, D.: Masking a compact AES S-box. Technical Report NPS-MA-07-002, Naval Postgraduate School (June 2007)

Steel, Cast Iron and Concrete: Security Engineering for Real World Wireless Sensor Networks

Frank Stajano, Dan Cvrcek, and Matt Lewis

Computer Laboratory, University of Cambridge
15 JJ Thomson Av, CB3 0FD Cambridge, UK
frank.stajano@cl.cam.ac.uk, dc352@cl.cam.ac.uk,
ml400@cam.ac.uk

Abstract. What are the real security issues for a wireless sensor network (WSN) intended to monitor the structural health of a suspension bridge, a subway tunnel or a water distribution pipe? Could an attack on the sensor network cause the structure to collapse? How easy is it for civil engineers or other domain experts to build a secure WSN using commercially available hardware and software?

We answer these questions by conducting a qualitative risk assessment with bridge and subway tunnel operators and by conducting penetration testing on commonly available commercial WSN hardware and software, namely the Crossbow MICAz motes running TinyOS and XMesh and communicating over IEEE 802.15.4.

1 Introduction

We are interested in the practical security of real-world deployments of wireless sensor networks (WSN). The goal of our project is to develop reliable ways of monitoring the condition of large civil engineering structures such as bridges, subway tunnels, water pipes and sewers so that structural damage (due for example to corrosion, materials fatigue, overloading or shifting of surrounding soil) may be noticed and remedied before the structure weakens to the point of failure.

Traditionally, such monitoring is effected through periodic visual inspection. However the cost of accessing the structure for inspection is high, not just in terms of the effort required for the people in hard hats to reach the part of the structure to be monitored (think of the main steel cables of a suspension bridge or the middle section of a 3 km stretch of subway tunnel between two stations) but also in terms of the downtime of the structure for its regular users. This means that, unless there is reason to suspect a problem at a particular spot, routine inspections are infrequent (perhaps every few months) and cannot easily pick up new faults as they develop. This motivates our search for a solution based on a fixed network of sensors—wireless rather than wired in order to facilitate deployment in hard-to-access locations. Within the project, we (the three authors of this article) are responsible for the security engineering aspects of the wireless sensor network.

From the security viewpoint, what is the problem to be solved? A preliminary challenge is to understand what damage could be caused by a failure (whether accidental or maliciously induced) of the sensor system. Would an attack on the sensor system just compromise the monitoring functionality, forcing the structure operators to fall back to manual inspection, or could it also have direct repercussions in the physical world, such as causing actual damage to the structure or to other nearby entities? If so, how? It is difficult to quantify such risks because there is essentially little or no prior experience of large civil engineering structures being monitored for many years by wireless sensors. Sometimes the structure operators themselves cannot imagine any serious security issues. It would be a mistake to launch into grand plans for encryption, key management and access control based just on what is fun for security researchers to do, rather than in response to recognised risks. So the consequences of security failures must be researched and understood, and the cost of security measures must be appropriate for the risks. The second challenge, assuming that the risk analysis requires and justifies such action, is to build a secure network of sensors using commercial off the shelf (COTS) hardware and software. Are current systems sold in a secure configuration? If not, is it still possible for non-experts to build a secure system out of such components? How? And how difficult is it?

In this paper we offer the following contributions. Firstly, we report on our initial qualitative risk assessment, carried out by interviewing the operating manager of a large suspension bridge and a contractor responsible for part of a large subway tunnel network (section 3). Secondly, and most significantly, we assess the practical security of the particular COTS system adopted by our team, the Crossbow MICAz motes running TinyOS or XMesh, together with the Stargate gateway: we design and implement a variety of attacks on this system and we report on the security problems we found, together with appropriate fixes where possible (section 4). As a further contribution to WSN security we ported the TinySec security library to the MICAz motes 4. While some of our attacks exploit generally known vulnerabilities, others like selective jamming (section 4.3) and power exhaustion through routing table manipulation (section 4.5) are original and interesting in their own right. In section 4.3 we also demonstrate how an attacker can undetectably alter messages in an IEEE 802.15.4 radio environment. Finally, based on the experience we gained, we offer some architectural recommendations (section 5), independent of the particular hardware and software we used, that will help future teams design and deploy more secure WSN systems out of COTS hardware and software.

2 Scenario

2.1 Sensing

The purpose of a sensor network in our scenario is to monitor ambient conditions for hints that the structure may be deteriorating. Here is a non-exhaustive list of examples of what we monitor and why.

¹ The source code is available under GPL from <http://www.winesinfrastructure.org/>

Bridges. The main cables of a 2 km suspension bridge are almost 1 m in diameter; each of them is actually a bundle of over ten thousand 5 mm steel wires, all anchored in strands into huge concrete blocks in underground chambers at either end of the bridge. We monitor temperature and humidity at various points in those chambers for signs of conditions that might lead to corrosion.

Some of the wires do break over the lifetime of the bridge owing to corrosion, defects and stress. A large safety factor is built in by design, to ensure that the main cables will still support the weight of the loaded bridge despite a number of inner wires having snapped; but monitoring the rate of breakages is very hard. A visual inspection, involving opening up the main cables with wedges, is extremely costly and disruptive; it will only spot breakages that are close to the inspection point and it may itself cause further breakages by stressing the cable. We will instead be using sensors based on acoustic monitoring of the “ping” sound made by a wire snap: with several synchronised sensors on the main cables, one may approximately locate the position of a breakage. Such sensors require a 100 kHz sampling rate, which imposes stringent performance constraints on the nodes.

Tunnels. A subway tunnel is essentially a hollow underground burrow whose walls are lined with large cast iron or concrete tiles. As the surrounding soil moves, settles and subsides, the cross-section of the tunnel deforms and the walls of the tunnel get damaged, perhaps developing cracks. We monitor existing cracks with sensors that measure the displacement across the edges of the crack. We also use inclinometer sensors to measure whether a given tile moves (by as little as a hundredth of a degree), indicating deformation of the tunnel that might lead to further cracks. We also monitor relative humidity, temperature and vibrations.

Water pipes. A large water distribution pipe can be almost 1 m in diameter. The opening and closing of valves causes pressure waves that stress the pipe and may eventually result in leaks. A leak in a large water pipe may discharge substantial amounts of water and cause a local flood in less than an hour. Prompt intervention is essential. We monitor water pressure at various points in the pipe and infer the presence and location of leaks with mathematical models.

2.2 Network Architecture

The standard architecture for this type of application is a three-tier wireless sensor network. At the bottom tier, the sensors are attached to nodes² that form a multi-hop ad-hoc network. Modern motes tend to use IEEE 802.15.4 as their communication standard³ but some older motes may use unlicensed frequencies such as 868 or 916 MHz.

At the middle tier, the data measured by the motes is routed to a gateway, physically located near the nodes (e.g. inside the tunnel) because of obvious

² Also known as *motes* from the seminal “Smart Dust” paper [1].

³ This standard is often colloquially indicated as ZigBee but, strictly speaking, most current motes do not implement the higher layers specified by ZigBee on top of the PHY and MAC defined by 802.15.4.

connectivity requirements. The gateway, usually an embedded Linux PC, may perform some preprocessing on the collected data. Communication between the gateway and the next tier is frequently implemented via GPRS, which is easy to deploy in isolated areas. However, GPRS is relatively slow and expensive and therefore may not be suitable for applications that require high data rates or always-on connectivity (as opposed to, say, overnight batch logging of a few data samples collected during the day). It also does not work in tunnels. Where ADSL is available, it is usually more convenient.

At the top tier, the data collected by the gateway(s) is aggregated into a database running on a central server (usually a PC) that will be accessed by various applications for visualisation and processing of the data.

3 Qualitative Risk Assessment

Since the value of our project to the security community comes primarily from its link to real-world installations, we started by interviewing two senior representatives of organisations that respectively operate a large suspension bridge and a system of underground tunnels. We wanted to elicit their perception of risks rather than relying on our own guesswork. This is an ongoing portion of the project and we next expect to interview a water distribution operator.

Direct consequences in the physical world. The disruptive potential of data modification or injection attacks is greatly amplified if the sensors are directly connected to actuators in a feedback loop (e.g. actuators close the valves as soon as a leak is reported). In such cases, altering bits in a radio message has a direct effect on the real world (a mains water pipe is closed down, stopping water delivery for a whole neighbourhood).

Our first round of questions to the bridge and subway operators therefore aimed to elicit whether the WSN would plausibly be linked to any actuators once the system reached maturity. For the bridge, the answer was a definite no: none of the measurements would have consequences requiring automatic and immediate action. Everything is reported to a control room from where human operators have direct visibility of the bridge. Any action, including such “soft” actions as activating road signs that lower the speed limit for motorists (something that is done in poor weather conditions such as high winds, frost and so on), is initiated manually by an operator who cross-checks the sensor readings with other clues (e.g. a windsock). For the subway tunnels the situation was more subtle, since operators cannot rely on a visual cross-check, but our contact could still not imagine a situation in which tunnel sensors would be directly attached to actuators without human intermediation. We are keen to interview a water operator, though, whose answer might be quite different in light of the leak scenario.

Confidentiality. We also asked whether the operators would be worried if sensor readings were not kept confidential. The bridge operator was not worried at all, since from his experience he did not expect any such reading ever to say anything

extraordinary or embarrassing. The tunnel operator was more sensitive and cautious: if a problem occurs he wants to hear about it first, and have a chance to address it before others (e.g. the press) know it occurred. A well designed WSN system should therefore be flexible; encryption should only be activated if confidentiality protection is worth more than its cost in energy consumption, key management and system complexity.

Short-Term Integrity. Concerning the threat of false negatives (attacker making us believe that there is no fault when there is one, thus stopping the structure owner from carrying out maintenance that might fix the problem, leading to aggravation of the problem with time and possibly partial or total collapse) we learnt from these interviews that, over the life of these structures up to this point (26 years for the bridge; 118 years for the underground tunnels), the number of serious structural incidents has been nil or very low. This implies that trying to take down the structure by hiding spontaneous faults that the sensor network would report is a strategy that might force the attacker to wait for a very long time! The situation might be different if the attacker also *caused* the damage, as well as was subsequently trying to hide it from the sensors, but if this were perceived as a serious threat then safeguards against causing physical damage would be much more of a priority than those against network tampering.

With the injection of false positives, instead, the sensors report damage (e.g. several wire snaps one after the other) that has not actually occurred. This forces the maintenance operators to waste time trying to locate and repair a non-existent fault, e.g. by opening up the main cable with wedges without finding anything. It may also force temporary closure of the structure while the problem is investigated. Therefore, false positives would be more disruptive, since unlike false negatives they could be triggered at will by the attacker. In any case the feedback was that any major alerts from the new system (WSN) would be viewed with some suspicion by the operators until the system had proved its worth. This was particularly true for the bridge operator, for whom the new system was a nice extra but not a necessity, whereas the subway operator had no “eyes” in the tunnel and was therefore more eager to accept and trust any technological development giving him greater monitoring power.

Medium-Term Integrity. The subway tunnel operators see value in analysing sensor data over the medium term (e.g. months) because they currently have no systems allowing remote monitoring of the state of the tunnels; and the tunnels themselves can only be physically inspected in the middle of the night when trains do not run. Systematic collection of data about the state of the tunnels would be very useful in budgeting for maintenance costs as it would allow more precise estimates of necessary works. Maintenance budget negotiations are currently based on very vague estimates derived from manual inspection of a small sample of accessible sections of the tunnels. Continuous monitoring would provide more reliable quantitative estimates that all parties involved would have an easier time accepting.

Long-Term Integrity. For structure operators, a significant perceived benefit of our automated sensing is the provision of decades-long monitoring logs that will

allow them to carry out previously impossible research on long term behaviour (and, inevitably, decay) of materials when plotted against influencing factors such as humidity, acidity, pollutants and stresses. The monitoring thus becomes particularly valuable when it allows reliable data collection over many years. One requirement is therefore that data be collected and stored in a sensibly designed non-proprietary format that can still be read after decades, under the assumption that any component of the physical implementation of the system will be replaced by something newer in due course. Another consequent requirement is on data integrity: corruption of sensor readings, especially if on a small enough scale as to go undetected at acquisition time, would invalidate the whole historical database and make the exercise useless. This requires an architecture in which, regardless of confidentiality, integrity of sensor readings is preserved at all costs.

Availability. We explained that it would be technically impossible to eliminate denial of service attacks on a wireless network and tried to understand their practical consequences for the operators. Apart from the financial loss of having wasted money on an unusable sensor system, even a complete jamming of the wireless network did not appear as a grave loss to the bridge operator, who did not expect ever to depend entirely on the WSN output; but it sounded somewhat more embarrassing for the subway operator, who anticipated his newly gained real-time “eyes” on potential cracks as a facility he would not want to do without.

Conclusions. The general conclusion from the interviews with the structure operators is that data integrity is the most important security property for this type of application. No direct link from sensors to actuators is envisaged in the two systems we discussed, so the main effects of an attack on the WSN are invalidation of collected data or incapacitation of the WSN system rather than damage to real world facilities. This is therefore not a very high risk setting. Having said that, there is still scope for attackers causing disruption to users of the structure insofar as denial of service or injection of false positives may lead the operators to close the bridge or the tunnel for safety reasons while the problem is investigated.

4 Attacks on a Real System

Inspired by the above-mentioned user concerns, we examined the available WSN technologies for vulnerabilities threatening the desired security properties. As we did that, we also found problems that the operators had not anticipated.

Our goal was to assess practical security of wireless sensor networks on a real, physical system, as opposed to just in theory or through simulations. So we targeted our attacks on the particular platform that our project adopted, namely the MICAz mote from Crossbow, running TinyOS v1.1 and XMesh from MoteWorks 2.0.F, together with the Stargate rev. 1.2 as a gateway.

A superficial reader might comment that, since we chose the components and assembled the system ourselves, any security holes we find only reflect on our own incompetence. On the contrary, the spirit of our investigation was to

imagine that a team of application experts (in this case civil engineers), assumed to be security-conscious but not security experts, puts together a system using COTS components, following the manufacturer's instructions and activating any recommended security features. We set out to assess the practical security of the resulting system and to suggest ways of improving it where appropriate.

Our limited budget and manpower would never have allowed us to carry out a comparative study of all commercially available WSN platforms to determine the most secure one, so that was never a goal. Nonetheless, we believe our results will be interesting for users of other platforms too.

Each of the attacks or exploits described in this section has been carried out and validated on actual hardware. We report sufficient details to convince the reader that a vulnerability exists and has been exploited by us, but stop short of supplying malicious readers with a cookbook. We also describe how to fix the problem wherever possible. As a courtesy we supplied a copy of a preliminary version of this paper to Crossbow in September 2007, to give them a chance to release security patches based on our advisories.

4.1 Our Platform

The TinyOS operating system runs on various hardware platforms including MICA2, MICAz, Iris (Crossbow Inc.), Tmote (Moteiv), Intel Mote, Intel Mote 2 (Intel). It is a modular system that allows easy extension with drivers for new sensor boards or functionality.

TinyOS v1.1 appears to be the most commonly used version in practice: v2 exists but is not stable yet. TinyOS may be deployed as is, or in conjunction with a commercial derivative such as XMesh from Crossbow or Boomerang from Moteiv. Being an open source project, TinyOS is reasonably easy to analyse for stability and security issues. TinyOS v1.1 has been stable since September 2003, so it can be considered a fairly mature product.

We began by analysing TinyOS, focusing our attention primarily on cryptography and routing protocols. The first big surprise was that, while TinyOS ships with the cryptographic module TinySec [2], the latter can only be compiled for MICA2 motes and there is no implementation available for the current generation of motes with 802.15.4-compliant radio chips. This means that all the networks based on modern Crossbow, Moteiv or iMote devices are vulnerable to a slew of attacks from even relatively unskilled attackers. To address this deficit, we ported TinySec [4] to make it run with the latest 802.15.4 chips—namely the Texas Instruments CC2420 chip, used in most of the motes mentioned above—so that we could test our attacks on cryptographically secured networks. In the process of performing this port, we noticed that the TinySec MAC generation code uses a fixed block size of 7 bytes, while the underlying block cipher has a fixed block size of 8 bytes. While we do not believe that this causes a security exposure, our port corrects this behaviour to use the same block size

⁴ See footnote [1].

as the underlying cipher⁵. Note that, as the TinySec authors themselves point out, TinySec provides no protection against replay attacks. Unfortunately this protection does not exist at any layer of the TinyOS radio stack either. In addition, since by design TinySec only supports a single key shared across the entire network, there is no protection against physical capture of one mote.

A general virtue (security-wise) of the TinyOS code is that it is very lean and spartan. The underlying system supplies minimal functionality, preferring to export simple primitives to applications. One of the results of this is that the basic TinyOS system seems to be fairly secure without making much apparent effort to be so; several times during the course of our analysis we tried out attacks that we thought might work, but were blocked by TinyOS' minimalist philosophy. In contrast to this, the commercial system we analysed, Crossbow XMesh, exports a very rich set of features to applications running on it and consequently exposes a much wider attack surface.

Concerning power consumption, a mote using its radio continuously would exhaust a pair of alkaline AA batteries in a couple of days (a couple of weeks for the expensive D lithium batteries we use). In normal use, with a duty cycle of around 1%, a mote lasts for several months.

4.2 Classes of Attacks

We chose not to concentrate on **physical attacks** ³ on the sensors and on the nodes attached to them, not because we think they are impossible⁶ but because an attacker with physical access to the sensors could with comparable effort stage much more destructive attacks on the structure itself, for example by using explosives. We therefore focus on attacks on the communication systems, primarily the ad-hoc radio used by the sensor nodes but also the back-end link from gateway to central server.

We studied three broad types of attacks: **data payload attacks** that change the content of data packets; **network attacks** that affect the functionality of the network, for example by preventing communication, taking down specific links, modifying the routing topology or rewriting the firmware of a node; and **system attacks**, potentially the most damaging, in which the attacker exploits a vulnerability in one part of the system architecture (e.g. the wireless network) to gain control of other parts (e.g. the gateway or the central computer).

Attack mechanisms we employed included jamming (at various degrees of selectivity and at different layers in the stack), replay attacks, packet injection or corruption (where the injected or malformed packets were specifically crafted to probe for vulnerabilities or to trigger known vulnerabilities) and ACK spoofing.

⁵ This divergence from a correct implementation will become a compatibility issue if end-to-end keys are used, as the MAC algorithm will have to be implemented on both the motes and the gateway. The NesC code for the motes cannot be compiled for the gateway, so its use of non-standard parameters hinders interoperability.

⁶ On the contrary, with so many unsupervised nodes over a large area, we believe one cannot exclude that a few nodes may at some point be physically captured, even though attackers are unlikely ever to be able to take over a *majority* of the nodes.

4.3 Jamming

While it is well known that no one can prevent physical jamming of the radio channel, appropriate design decisions can reduce a system's vulnerability to denial of service (DoS) attacks and increase the cost of such attacks. In this section, we investigate the baseline case of an attacker using an unmodified COTS mote as the transmitter. It is understood that a more resourceful attacker, with a higher power transmitter, could cause greater damage.

Description. We used a MICAz mote to jam the communication between other motes by transmitting at the same time as them, thereby causing a collision. Unlike most previous jamming attacks, ours is selective and jams only a subset of the packets (e.g. those coming from a specified victim) while leaving others unaffected. The attack can therefore be used to selectively disconnect individual motes or whole regions from the network. It is also hard to detect because unaffected nodes do not notice anything unusual.

The algorithm is as follows:

1. We select criteria for messages to be jammed, e.g. `senderID = 3` and `messageType = AM_MULTIHOP`.
2. We compile the code with the selected criteria into the attacking mote and deploy the mote in the vicinity of the victim.
3. When the attacking mote detects a transmission, it listens to just enough of the message to determine whether the packet meets its jamming criteria.
4. If the message meets the criteria, go to step 5. In our example, we have a match if `byte 13 = 3` and `byte 9 = AM_MULTIHOP`. Loop back to step 3 if the criteria were not met.
5. The attacking mote switches the radio chip to transmit mode and jams the rest of the message by transmitting on the same channel.

Steps 3 and 5 are difficult for technical reasons. To start with, the CC2420 is a packet based radio, so in normal operating conditions the microcontroller is only informed of radio activity after a complete packet has arrived. This obviously makes step 3 difficult since, by the time we can tell that we should jam a packet, it has already been transmitted. We were able to overcome this difficulty by putting the radio chip into a debugging mode, where each bit received from the radio was sent to one of the microcontroller's input pins as it arrived.

Our second problem in implementing the attack arose from the tight timing requirements imposed by the data rate of the radio—250 kbps. The ATMega128 microcontroller is clocked at 7.37 MHz, effectively giving us 30 instructions to process each bit. This would not be too much of a problem were it not for the fact that invoking a hardware interrupt appears to take around 30 clock cycles, meaning that our interrupt handler (which gets invoked once per bit) was always too slow. However, having the first invocation of the interrupt handler busy-wait for the remaining bits in the frame allowed us to meet the timing requirements.

Finally, switching from receive mode to transmit mode takes some time, so if the frame was particularly short we would frequently miss the end of the packet by the time we started jamming. By iteratively optimising the code, we were able to reliably jam frames with as few as 5 bytes of data.

Empirical Results. The 802.15.4 standard implemented by the motes uses direct-sequence spread spectrum (DSSS) to increase resilience to noise and jamming. However, since 802.15.4 specifies the spreading code to use, our motes shared the spreading code with the target, thereby nullifying the protection offered by DSSS. We carried out several experiments to measure the success rate of jamming. The experiments used three motes: transmitter, receiver, and jammer. The transmitter sent out a packet every 200 ms and the receiver, connected to a laptop, forwarded the received messages to a laptop where they were logged. All three motes were positioned 45 cm above the ground.

The jamming appeared successful in an open space, but with some anomalies. All messages were jammed when the attacking mote was between the sender and the receiver. The initial results (see Figure 1) suggested that the angle defined by transmitter-receiver-jammer had a significant impact on the success of the jamming, but later experiments showed a lack of repeatability, with variations between 30 and 88% for a given position. Complete jamming (100% of frames jammed) was achieved in several configurations of the three motes.

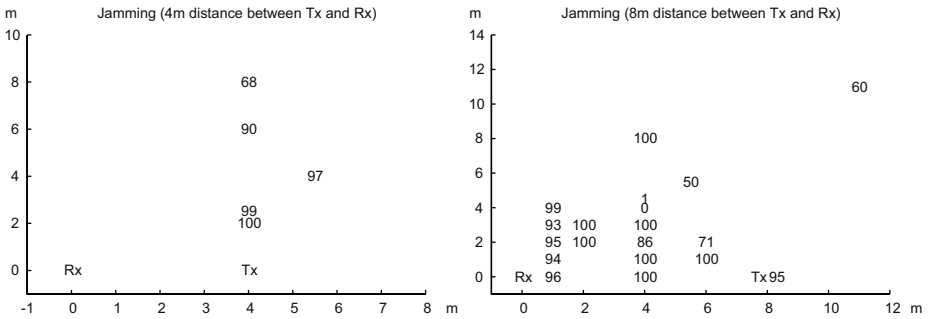


Fig. 1. Success rate of jamming depending on the position of the attacking mote. The Tx and Rx labels are the transmitting and receiving motes. The numbers 0–100 in the graphs denote the percentage of packets that were jammed when the attacking mote was in that 2D position relative to Tx and Rx (section 4.3).

We tried to find out why our results were so unpredictable, especially when the motes were close to the ground. A radio engineer told us that this is due to interference between the direct ray and the ray reflected from the ground, as the path difference between the two rays can be close to a half wavelength when the motes are on the ground. This results in destructive interference between the two rays, causing a weak signal. We think it may be some kind of voodoo.

Risks. The ability to selectively jam frames allows an attacker to:

- Make specific sensors deaf and/or mute;
- Prevent any sensor data from reaching the database, by jamming close to the gateway;
- Perform a man-in-the-middle attack by having a second mote listen for jammed frames and resend chosen ones with altered data.

The first two of these are simple DoS attacks and so may be written off by some as unimportant. The ability to alter the contents of any message while remaining undetected is rather more serious, as it gives the attacker complete control over the output of the network and therefore indirect control over all the systems controlled by the data reported by the sensor network.

Since the jammed frames are silently dropped by the radio chip, TinyOS is never notified. As such, a mote is unable to determine that a jamming attack is taking place unless its radio driver has been specifically modified to run in debug mode in order to detect this.

Underlying Cause. The vulnerability of a network to jamming as described here is simply a result of communicating over a shared medium to which adversaries have access (in this case radio).

Fix. To make it more difficult for an adversary to perform this type of selective jamming attack, the frame header could be moved to the end of the frame. This would mean that by the time an attacking node had decided to jam a frame, it would be too late. Unfortunately this defence is not perfect and succumbs to an attacker with two motes: one to jam every frame and one to resend any frames that do not meet the jamming criteria⁷. It is also expensive in terms of battery life, as each node must listen to and buffer the whole packet, instead of just the header, before knowing whether it was destined for itself. Detection of this type of jamming could be achieved by having one or more “observer” motes running in debug mode and listening for suspicious patterns of corrupted frames.

4.4 Counter Overflow Attack

Description. Another attack that can be mounted against a wireless network is a replay attack. This type of attack works even against an authenticated and encrypted network. One of the simplest protections against it is to use monotonically increasing counters to ensure the freshness of messages. TinyOS and TinySec use a 16 bit counter to distinguish messages, with TinySec using another counter as part of the encryption initialisation vector. Although TinySec explicitly does not offer replay protection, even if it did, the attack presented in this subsection would still work, unless TinySec increased the length of the counter. The network stack decides whether to accept a received message as valid according to the value of the counter in the message⁸. If the counter in the message is higher than the stored counter, the message is accepted. If it is lower but the difference is within an allowed range, the message is also accepted. Any

⁷ We are not discussing all the details related to the necessity for the attacker to learn the whole content of every packet it jams in order to retransmit it if needed. This can be achieved in several ways, including jamming a part of the message that the attacker can reconstruct (such as the CRC, provided that there were no real errors), or jamming the message twice in different places, counting on the fact that the sending node will retransmit—although this latter strategy would be less stealthy.

⁸ This is also used to compute the number of missed messages and subsequently the quality of the link with that neighbour.

other value causes the message to be rejected and the stored counter value to be zeroed. Each mote tracks the counter values of all its neighbouring motes.

For a replay attack to work, the receiving node needs to be ready to accept a message with the same counter value as the one we eavesdropped and intend to replay. We can achieve this by having the recipient's counter overflow and wrap around to the desired value. We thought of at least three methods to cause a counter overflow. First, we can inject fake messages that the target mote will forward, causing the target's counter to increment once per message we send. Second, we can use the routing table attacks described in section 4.5 to create a loop in the network: this causes a packet storm which eats up counter values. Finally, we can simply jam message acknowledgements from the target's parent as this will cause the target to retransmit each message several times.

Note that, if XMesh receives a message whose counter is set to zero, it bypasses the routing mechanisms used to verify the freshness of the message, obviating the need to overflow the counter. In addition, the receiving mote will not increment its stored counter, which means that the message has no effect on the link quality calculations. An attacker might exploit such "features".

Risks. When the message counter overflows we may initiate a replay attack with previously eavesdropped messages, even if these messages feature cryptographic integrity protection. This allows us to inject false negatives, false positives, or just slightly corrupted data to invalidate the long term monitoring. It also allows us to manipulate routing messages, even if they are authenticated. The message counters form the basis of many cryptographic mechanisms that rely on freshness, particularly in the absence of timestamps. This vulnerability would affect security mechanisms built from these cryptographic primitives.

Underlying Cause. Our three counter overflow methods rely respectively on the lack of source authentication, on the possibility of routing table manipulation (section 4.5), and on jamming (section 4.3). As such, the underlying causes of the counter overflow attack are the union of the underlying causes for these sub-attacks.

4.5 Routing Table Manipulation

Description. Nodes in an XMesh network populate their routing tables by listening to periodic broadcasts issued by their neighbours, which contain:

- The node's current parent;
- The path cost of sending a message from the node to the base station;
- A list of some of the node's neighbours and an estimate of how well it can receive messages from each.

When a node receives such a message from a neighbour, it updates its internal table of neighbours, which it consults every few minutes⁹ to choose a new parent. All future messages are routed through this parent. The new parent appears to

⁹ Every 30 seconds if the node is disconnected.

be chosen to minimise the path cost of sending messages back to the base station, with the link quality to each potential parent being a factor in the cost of the hop to the parent. The routing algorithm may thus be thought of as a distributed single-source shortest path algorithm.

Since all communication in the XMesh network is unauthenticated and unencrypted, an attacker can interfere with routing messages and thereby change the topology of the network to suit her whims. We created a Python module called `wsn` to read, modify, create and inject messages into the sensor network and we used it to implement and validate these routing attacks.

Risks. The ability to choose a topology for the network gives the attacker a lot of power. For example, she can cause all the traffic in the network to be routed through a mote she controls, allowing her to selectively drop or modify any message. Alternatively, she can cause a direct attack against the motes themselves by creating routing loops where, e.g. one route is $2 \mapsto 3 \mapsto 2 \dots$. In such a configuration, nodes 2 and 3 will send messages back and forth, rapidly consuming their batteries. Using this method, we have been able to increase the peak rate at which nodes send messages to over 300/s with a mean value of around 25 messages per second. This can be used for a very powerful “sleep deprivation torture” [4] attack.

There is a low power implementation of XMesh that puts radio and micro-controller to sleep and they wake up only when necessary. However, any mote forwarding for at least one of its neighbours must wake up the radio chip regularly. The XMesh low power mode wakes up the radio chip roughly every 125 ms and it listens for a period of 1 ms. This means that the duty cycle is around 0.8%. The attack increases this value substantially to tens of percents.

Karlof and Wagner [5] discuss other routing attacks.

Underlying Cause. The problem here seems to be that the topology of the network is decided by the motes themselves while they do not have enough trusted information about the physical structure of the whole network to make sensible decisions. The ad hoc nature of the network appears to cause significant security issues, as everything received is trusted.

Fix. A natural solution to such trust issues would be to cryptographically authenticate messages sent by legitimate nodes. We could, for example, have all the nodes in the network share a key and use TinySec to authenticate and encrypt all messages sent over the network. This would mean that an attacker would not be able to forge routing messages without knowledge of the key. One problem with this scheme is that physical capture of a single node would reveal the network key and render the entire network susceptible to attack. Another problem is that authentic routing messages from one part of the network may still be recorded and replayed in other parts of the network, causing routing anomalies. We could stop this type of attack by using per-link keys (unique keys shared by each pair of neighbours), at the cost of complicating the key management process.

No cryptographic fix, however, would stop what has been called a “flagpole” attack, in which the attacker moves a victim mote up and down a flagpole in order to make other motes waste their battery updating their routing tables.

4.6 Over-the-air Programming

Description. One of the optional features that can be compiled into the motes making up an XMesh network is over-the-air programming (OTAP). This is a mechanism by which an entire network of motes can be reprogrammed remotely by sending them the code to execute. If OTAP is enabled on a network, no authentication is required to request a reprogram; an attacker able to send traffic to the network can therefore cause every mote to execute code of her choice.

Risk. The ability to reprogram motes allows an attacker to entirely control all data sent by the network. This means that she would be able to choose whether to send data from the sensors, what the apparent readings of the sensors were and how often the readings were reported.

Underlying Cause. The underlying problem here seems to be that there is no concept of authentication between any parties communicating over the network. As a result, all traffic is trusted and this is clearly a bad transport over which to run critical protocols such as OTAP.

Fix. The obvious action to mitigate this risk would be not to use OTAP. One fix could be to authenticate OTAP messages, e.g. with a scaled down version of the CMS standard for secure firmware update (RFC 4108). Using public key cryptography, generally thought of as computationally infeasible on low-end hardware, might be acceptable for such rare events as authenticating the signature of an OTAP request. Then the network would not be vulnerable to the physical capture of one node and so OTAP could be used securely.

4.7 Remote Command Execution in XServe

Description. XServe is a middleware component that connects the WSN to the back-end. If XServe is started with the `-h` flag, it starts a web server on a user-specified port. This web server is intended to be used to display the output of the attached sensor network. Unfortunately, one of the scripts supplied with the web server contains a bug that can be exploited by an attacker to execute arbitrary commands on the computer running XServe.

Risk. XServe may be run on a Stargate gateway or on the top-tier server. If the web server were enabled on a Stargate and an attacker could access the web server port, she could gain shell level access to the Stargate, obtaining complete control (read, write, modify, drop) over all the data sent from the sensor network to the central servers. If the web server were running on a central database server, an attacker could gain access to the administrative network, potentially gaining complete control over all new data from the network, as well as the ability to modify historical data and attack other parts of the network.

Underlying Cause. This exposure results from a simple programming error, but the underlying problem is really that the XServe component is far too feature rich. From a security viewpoint it is hard to defend the decision of equipping it with a built-in web server, for example.

Fix. We can fix this vulnerability by patching the script or by not using XServe's built in web server. However, since software errors cannot be eliminated, we suggest running complex software on safely stored data, outside the critical parts of the system.

4.8 Stargate Unhardened to IP Attacks

Description. The Stargate we bought new in 2007 shipped with very old versions of several pieces of software, many of which contained exploitable vulnerabilities. Some of the more serious issues are summarised in Figure 2.

Component	Exposure	CVE Reference
OpenSSH	Local root	CVE-2002-0083
		CVE-2003-0682
OpenSSH	Remote root	CVE-2003-0693
		CVE-2003-0695
Kernel	Remote root	CVE-2004-1137
Kernel	Local root	CVE-2005-1263
Kernel	Local root	CVE-2004-1235
PostgreSQL	Remote shell	CVE-2005-0247
PostgreSQL	Remote shell	CVE-2005-0245
PostgreSQL	Remote shell	CVE-2003-0901

Fig. 2. Some vulnerabilities affecting the Stargate. CVE reference numbers may be resolved at <http://cve.mitre.org/cve/>. (Section 4.8).

In addition to these problems caused by outdated software, the Stargate ships in an insecure configuration: it has a default, weak root password and an SSH daemon installed. In addition, if the PostgreSQL database is installed on the Stargate (a recommended and supported configuration), a database super-user is added with a default, weak password. The documentation does not explain how to change these passwords, nor does it suggest that you do. Furthermore the existence of the database super-user account is not even mentioned.

Risks. A remote attacker with access to the Stargate's IP interface may be able to gain root access or crash the Stargate. This would result in either a complete compromise of the sensor network or a complete DoS of the network.

Underlying Cause. The Stargate was not designed for security, as demonstrated by the outdated software, weak passwords, lack of security related documentation and unnecessarily high number of services running.

Fix. Patch, configure, minimise. If possible, update the software on the Stargate, in particular OpenSSH, PostgreSQL and the kernel. Do not run `sshd` on the Stargate and drop all inbound IP traffic to the Stargate that is not already part of an established TCP stream. This implies that all connections to the back-end network would have to be initiated by the gateway. In addition, change passwords for the system root account and the `tele` user in the PostgreSQL system.

5 Architectural Recommendations

Securing the Gateway. The WSN gateway is a bottleneck through which all data related to the sensor network flows. It is therefore crucial that it be properly secured. The usual configuration has the gateway performing many complex operations on the received data and exporting several services to the IP network. We instead recommend not to offer any services to the IP network (e.g. no SSH or HTTP daemons) and not to perform any analysis of the sensor data on the gateway. The gateway should act only as an SSH *client*, rather than server, and simply relay data from the WSN to the server. This would significantly reduce the attack surface of the gateway, resulting in greater security.

Key Management. Clearly, TinySec's default key management approach with a single key for all devices is inadequate to protect an unattended WSN, as the capture of a single node would expose the entire network. The use of pairwise link keys between motes would not protect against attacks on the gateway and another cryptographic protection would have to be used for communication with the central server. Based on the risk assessment in section 3, we recommend the use of pairwise end-to-end keys between each node and the central server to protect the integrity of the sensor data and to provide source authenticity. Insofar as routing is crucial to the security of the network, based on the vulnerabilities exposed in section 4.5 we also recommend the use of pairwise end-to-end keys between each node and the gateway to protect routing table data.

Routing. Many papers on WSNs axiomatically accept the smart-dust-derived assumption that individual nodes (possibly dropped from an aircraft) know nothing about each other's position and that therefore the network can only be built in an ad hoc, decentralised fashion. But, in our scenario, the deployment engineers *know* where they want to place the nodes—for example where they see cracks that need monitoring. We therefore suggest taking advantage of that positional information, perhaps by precomputing an initial (though sub-optimal) set of routing tables and preloading it into the nodes. We also suggest that routing calculations be performed centrally, for example at the gateway, after collecting authenticated local connectivity information from the nodes, since visibility of the whole connectivity graph would allow for the discovery of a better global solution. We are currently working on such a system.

Reviewing the risk assessment. Since the monitoring operation might last for several years, the risk assessment should be repeated at regular intervals to ensure that it still reflects the current usage patterns. It is common for systems to be used in ways that were not originally envisaged, so the protection goals and consequent security measures should be kept up to date.

6 Related Work

Although very many papers have been written about wireless sensor networks, experience papers reporting on real-world deployments are a minority: they include at least Mainwaring et al [6] who monitor seabird nesting environment and

behaviour, Arora et al [7] who deploy a perimeter control WSN and Werner-Allen et al [8] who monitor an active volcano. Closer to our scenario are Krishnamurthy et al [9] who monitor equipment for early signs of failure and especially Kim et al [10] who monitor the structural health of the Golden Gate bridge.

Similarly, although there is a vast literature on security of WSN, including such milestones as Perrig et al [11] on efficient broadcast stream authentication, Eschenauer and Gligor [12] on random key predistribution, Hu et al [13] on secure routing and Chen et al [14] on energy efficient topology maintenance, few such papers deal with attacks on actual sensor network platforms. One notable exception is the brilliant work by Becher et al [3] on physical attacks, providing concrete data on the effort required to extract secrets from a mote. Closer to our own investigations on 802.15.4 jamming are Wood et al [15], who discuss jamming attacks and countermeasures in detail, although their focus is on energy efficiency for the attacker rather than on being able to target a specific victim selectively. The TinySec work of Karlof et al [2] is of particular significance since it resulted in running code which we used extensively. One promising effort in a similar vein is that of Luk et al [16] for which, however, the code was only released after we completed this paper.

Our specific interest lies at the intersection of these two sets: real-world WSN deployments and real-world WSN attacks; so far we have not been able to find other papers in this subset. We believe that practical security is a field worth exploring in greater detail before proceeding with actual deployments.

7 Conclusions

What are the risks of a WSN that monitors large engineering structures? In the situations we examined, the sensors are never linked to actuators; therefore, deploying a WSN in such circumstances does not introduce major new risks. Typically, the worst outcome of an attack is that data gathered from the WSN will be useless, not that the structure itself will be directly endangered. Of course, if the WSN or the collected data becomes useless, there is a financial loss; moreover, false alarms might cause secondary losses through downtime; so, ensuring the integrity of the WSN is still a worthy goal. Considering our qualitative risk assessment (section 3), all the attacks presented in section 4 are potentially relevant, because they can be used directly or indirectly to corrupt integrity of sensor data—the main concern of the structure owners as expressed during interviews. The implementation of countermeasures, which must crucially protect the whole system including back-end and gateway as opposed to just the motes, could be prioritised based on a quantitative risk assessment that established the likelihood of each attack.

Will a WSN built by well-intentioned and security-minded application experts be secure by default? Based on the components we examined, no. For a start, if users of MICAz motes (or any other motes using 802.15.4) wanted to “turn on the crypto”, they would find that TinySec does not even compile for their platform. Porting the code is not a trivial endeavour; fortunately, we have

now done it for them. Besides that, we have found and exploited a variety of vulnerabilities. The most devastating in practice are probably the most trivial from an intellectual viewpoint: the Over-The-Air Programming vulnerability (section 4.6), which allows an attacker to reprogram motes at will, and the Remote Command Execution vulnerability (section 4.7), which allows an attacker to gain complete control of the gateway or even the central server. Until such features are patched or protected, it is advisable to keep them disabled.

Among our other attacks, the most significant for the security researcher is probably our sleep deprivation torture attack based on routing table manipulation (section 4.5): it is crippling and very efficient because the attacker can set it up and walk away, as opposed to having to keep talking to the victims to drain their batteries out. Our selective jamming (section 4.3), too, is of interest as it is undetectable by OS and applications and can be used as the basis for more sophisticated attacks including packet rewriting (often assumed possible but rarely demonstrated in a modern ad-hoc radio context) and man-in-the-middle.

We trust that this paper will help vendors and users strengthen the security of their real-world systems.

Acknowledgements

We thank EPSRC for funding this work as part of project EP/D076870/1¹⁰ and the Isaac Newton Trust for co-sponsoring Matt as a UROP summer student. Dan also thanks the MSM 0021630528 project. We are grateful to our industrial partners, interviewed for the risk assessment, and to our colleagues from the Engineering Department of the University of Cambridge for support, especially Neil Houlton who also commented on the paper.

References

1. Kahn, J.M., Katz, R.H., Pister, K.S.J.: Next century challenges: Mobile networking for “smart dust”. In: Proc. 5th ACM MobiCom, pp. 271–278. ACM Press, New York (1999)
2. Karlof, C., Sastry, N., Wagner, D.: Tinysec: A link layer security architecture for wireless sensor networks. In: Proc. 2nd SenSys, pp. 162–175 (2004)
3. Becher, A., Benenson, Z., Dornseif, M.: Tampering with motes: Real-world physical attacks on wireless sensor networks. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 104–118. Springer, Heidelberg (2006)
4. Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Malcolm, J.A., Christianson, B., Crispo, B., Roe, M. (eds.) Security Protocols 1999. LNCS, vol. 1796, pp. 172–194. Springer, Heidelberg (2000)
5. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. Ad Hoc Networks 1(2–3), 293–315 (2003)

¹⁰ <http://www.winesinfrastructure.org/>

6. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proc. 1st ACM WSNA, pp. 88–97. ACM Press, New York (2002)
7. Arora, A., Ramnath, R., Ertin, E.: Exscal: Elements of an extreme scale wireless sensor network. In: Proc. 11th IEEE RTCSA, pp. 102–108 (2005)
8. Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., Johnson, J., Ruiz, M., Lees, J.: Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing* 10(2), 18–25 (2006)
9. Krishnamurthy, L., Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Kushalnagar, N., Nachman, L., Yarvis, M.: Design and deployment of industrial sensor networks: Experiences from a semiconductor plant and the north sea. In: Proc. 3rd SenSys, pp. 64–75. ACM Press, New York (2005)
10. Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., Turon, M.: Health monitoring of civil infrastructures using wireless sensor networks. In: Proc. 6th IPSN, pp. 254–263. ACM Press, New York (2007)
11. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: Spins: Security protocols for sensor networks. *Wireless Networks* 8(5), 521–534 (2002)
12. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proc. 9th ACM CCS, pp. 41–47. ACM Press, New York (2002)
13. Hu, Y.C., Perrig, A., Johnson, D.B.: Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wireless Networks* 11(1-2), 21–38 (2002)
14. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *wireless networks* 8(5), 481–494 (2002)
15. Wood, A.D., Stankovic, J.A., Zhou, G.: Deejam: Defeating energy-efficient jamming in ieee 802.15.4-based wireless networks. In: Proc. 4th IEEE SECON, pp. 60–69. IEEE Computer Society Press, Los Alamitos (2007)
16. Luk, M., Mezzour, G., Perrig, A., Gligor, V.: Minisec: a secure sensor network communication architecture. In: Proc. 6th IPSN, pp. 479–488. ACM Press, New York (2007)

Traceable Privacy of Recent Provably-Secure RFID Protocols

Khaled Ouafi^{1,*} and Raphael C.-W. Phan^{2,**}

¹ Laboratoire de sécurité et de cryptographie (LASEC)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

`khaled.ouafi@epfl.ch`

² Electronic & Electrical Engineering
Loughborough University
LE11 3TU, United Kingdom
`r.phan@lboro.ac.uk`

Abstract. One of the main challenges in RFIDs is the design of privacy-preserving authentication protocols. Indeed, such protocols should not only allow legitimate readers to authenticate tags but also protect these latter from privacy-violating attacks, ensuring their anonymity and untraceability: an adversary should not be able to get any information that would reveal the identity of a tag or would be used for tracing it. In this paper, we analyze some recently proposed RFID authentication protocols that came with provable security flavours. Our results are the first known privacy cryptanalysis of the protocols.

Keywords: RFID, Privacy, Untraceability, Authentication protocols.

1 Introduction

Radio frequency identification (RFID) tags are being deployed in many consumer, financial and governmental applications, for instance respectively in supply chain [1, 6, 20, 21, 32], in contactless credit cards [13], and in e-passports [15, 5, 14, 17, 22].

In view of the pervasiveness and inconspicuous nature of these tiny RFIDs, privacy for RFID tag users is a major concern that could potentially impede the public's long-term adoption of RFID-enabled applications. To the best of our knowledge, formal treatments of privacy for RFID protocols include the work of Avoine [2], Juels and Weis [16], Le, Burmester and de Medeiros [18]; and Vaudenay [34, 35, 26]. The difference in these models lie basically in the power of the adversary's tag-corruption ability.

We analyze in this paper the privacy (or security when relevant) issues of the following provably-secure RFID authentication protocols: the protocol by Lim

* Supported by a grant of the Swiss National Science Foundation, 200021-119847/1.

** Work done while the author was with the Laboratoire de sécurité et de cryptographie (LASEC), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.

and Kwon [19] at ICICS '06, and two protocols at AsiaCCS '07 by Le et al. [18]. The first protocol is a nice unconventional design in the sense that it achieves both forward and backward untraceability in the face of tag corruption, while typical protocols only provide backward untraceability. That paper also defined a provable security model for backward and forward untraceability. The latter two protocols are interesting since they come with provable security in the sense of universal composability [4] which has strong guarantees. In fact, except for subsections 4.2 and 4.3 corresponding to breaks on forward privacy/security and therefore the notion of tag corruption is inevitably assumed by definition, our attacks do not even need the strong requirement of corrupting tags [33, 16, 34, 19, 18, 35].

2 RFID Privacy Models

For completeness and for better clarity, we describe here the general untraceable privacy (UPriv) model [25] that will be the setting in which we use in later sections to demonstrate how to trace tags and thus show that the schemes do not achieve the notion of untraceable privacy.

In fact, the model defined herein can be seen as an alternative definition of the Juels-Weis model [16] in a style more in line with the Bellare et al. [3] models for authenticated key exchange (AKE) protocols, for which RFID protocols can be seen to have close relationship with. With this model as a reference, our emphasis throughout this paper is on the analysis of the privacy (or security) issues of recent RFID protocols.

A protocol party is a $\mathcal{T} \in \text{Tags}$ or $\mathcal{R} \in \text{Readers}$ interacting in protocol sessions as per the protocol specifications until the end of the session upon which each party outputs **Accept** if it feels the protocol has been normally executed with the correct parties. Adversary \mathcal{A} controls the communications between all protocol parties (tag and reader) by interacting with them as defined by the protocol, formally captured by \mathcal{A} 's ability to issue queries of the following form:

Execute($\mathcal{R}, \mathcal{T}, i$) **query**. This models *passive* attacks, where adversary \mathcal{A} gets access to an honest execution of the protocol session i between \mathcal{R} and \mathcal{T} by eavesdropping.

Send(U_1, U_2, i, m) **query**. This query models *active* attacks by allowing the adversary \mathcal{A} to impersonate some reader $U_1 \in \text{Readers}$ (resp. tag $U_1 \in \text{Tags}$) in some protocol session i and send a message m of its choice to an instance of some tag $U_2 \in \text{Tags}$ (resp. reader $U_2 \in \text{Readers}$). This query subsumes the TagInit and ReaderInit queries as well as challenge and response messages in the Juels-Weis model.

Corrupt(\mathcal{T}, K) **query**. This query allows the adversary \mathcal{A} to learn the stored secret K' of the tag $\mathcal{T} \in \text{Tags}$, and which further sets the stored secret to K . It captures the notion of *forward security* or *forward privacy* and the extent of the damage caused by the compromise of the tag's stored secret. This is the equivalent of the SetKey query of the Juels-Weis model.

Test_{UPriv}(U, i) query. This query is the only query that does not correspond to any of \mathcal{A} 's abilities or any real-world event. This query allows to define the indistinguishability-based notion of *untraceable privacy* (UPriv). If the party has accepted and is being asked a Test query, then depending on a randomly chosen bit $b \in \{0, 1\}$, \mathcal{A} is given \mathcal{T}_b from the set $\{\mathcal{T}_0, \mathcal{T}_1\}$. Informally, \mathcal{A} succeeds if it can guess the bit b . In order for the notion to be meaningful, a Test session must be *fresh* in the sense of Definition 2.

Definition 1 (Partnership & Session Completion). We say that a reader instance \mathcal{R}_j and a tag instance \mathcal{T}_i are partners if, and only if, both have output $\text{Accept}(\mathcal{T}_i)$ and $\text{Accept}(\mathcal{R}_j)$ respectively, signifying the completion of the protocol session.

Definition 2 (Freshness). A party instance is fresh at the end of execution if, and only if,

1. it has output Accept with or without a partner instance,
2. both the instance and its partner instance (if such a partner exists) have not been sent a Corrupt query.

Definition 3 (Untraceable Privacy (UPriv)). UPriv is defined using the game \mathcal{G} played between a malicious adversary \mathcal{A} and a collection of reader and tag instances. \mathcal{A} runs the game \mathcal{G} whose setting is as follows.

Phase 1 (Learning): \mathcal{A} is able to send any Execute , Send , and Corrupt queries at will.

Phase 2 (Challenge):

1. At some point during \mathcal{G} , \mathcal{A} will choose a fresh session on which to be tested and send a Test query corresponding to the test session. Note that the test session chosen must be fresh in the sense of Definition 2. Depending on a randomly chosen bit $b \in \{0, 1\}$, \mathcal{A} is given a tag \mathcal{T}_b from the set $\{\mathcal{T}_0, \mathcal{T}_1\}$.
2. \mathcal{A} continues making any Execute , Send , and Corrupt queries at will, subjected to the restrictions that the definition of freshness described in Definition 2 is not violated.

Phase 3 (Guess): Eventually, \mathcal{A} terminates the game simulation and outputs a bit b' , which is its guess of the value of b .

The success of \mathcal{A} in winning \mathcal{G} and thus breaking the notion of UPriv is quantified in terms of \mathcal{A} 's advantage in distinguishing whether \mathcal{A} receives \mathcal{T}_0 or \mathcal{T}_1 , i.e. it correctly guessing b . This is denoted by $\text{Adv}_{\mathcal{A}}^{\text{UPriv}}(k)$ where k is the security parameter. In relation to other models, note that the Le-Burmeister-de Medeiros model [18] similarly allows the corruption of tags. For the purpose of our attack descriptions in later subsections 4.2 and 4.3, it suffices to consider their definition of corruption in their model. This will be treated later as required.

The Vaudenay model [34, 35] is stronger than both the Juels-Weis and Le-Burmeister-de Medeiros models in terms of the adversary's corruption ability. In

more detail, it is stronger than the Juels-Weis model in the sense that it allows corruption even of the two tags used in the challenge phase. It is stronger than the Le-Burmester-de Medeiros model in the sense that it considers all its privacy notions even for corrupted tags, in contrast to the Le-Burmester-de Medeiros model that only considers corruption for its forward privacy notion.

We chose to describe our tracing attacks in later sections with reference to a defined model in order for more uniformity between similar attacks on different RFID protocols, and for better clarity to illustrate how an adversary can circumvent the protocols using precise types of interactions that he exploits, as captured by his oracle queries. This will facilitate the task of a designer when an attempt is made to redesign a protocol which had been attacked.

3 A Backward and Forward Untraceable Protocol

At ICICS '06, Lim and Kwon [19] proposed an RFID protocol that offers untraceable privacy (UPriv) both before and after corruption of a tag. This is indeed a major feat, since other RFID schemes in literature are only able to treat backward untraceability, i.e. a corrupted tag cannot be linked to any past completed sessions.

The initialization phase is as follows:

1. The reader chooses a random secret K_i for each tag \mathcal{T}_i , and evaluates $m - 1$ evolutions of $K_i^0 = K_i$, i.e. $K_i^j = g(K_i^{j-1})$ for $1 \leq j \leq m - 1$, where g is a pseudorandom function. It then computes $t_i^j = ext_{l_2}(K_i^j)$ for $0 \leq j \leq m - 1$, where l_2 is some appropriate bit length, $ext_l(x)$ is an extraction function returning l bits of x .
2. The reader also chooses a random u_i for each tag \mathcal{T}_i and computes a key chain $\{w_i^j\}_{j=0}^{n-1}$ of length n , such that $w_i^n = u_i$ and $w_i^j = h(w_i^{j+1})$ for $0 \leq j \leq n - 1$, where h is a pseudorandom function.
3. The tag stores $\langle w_{i,T}, K_i \rangle$ where $w_{i,T} = w_i^0$ and initializes a failure counter $c_i = 0$.
4. The reader creates two tables L_1, L_2 for \mathcal{T}_i in its database, where L_2 is empty and L_1 has entries of the form $\langle s_i, \{t_i^j\}_{j=0}^{m-1}, u_i, n_i, w_{i,T}, w_{i,S} \rangle$ where $n_i = n$ and $w_{i,S} = w_i^1$ thus $w_{i,T} = h(w_{i,S})$.

After initialization, a normal protocol session is illustrated in Fig. 1, where f is a pseudorandom function. For further discussions on this protocol, the interested reader is referred to [19].

Tracing the Tag. For the purpose of understanding our attack, it suffices to review the gist of the Lim-Kwon protocol. The tag updates its stored secret K_i in two possible ways. If the reader is successfully authenticated, it would update as $K_i = g(K_i \oplus (w_{i,T} || r_1 || r_2))$. Else, the tag would update as $K_i = g(K_i)$, up to m times of unsuccessful authentications, after which the tag stops updating its K_i . This eventual non-updating allows the reader to catch up.

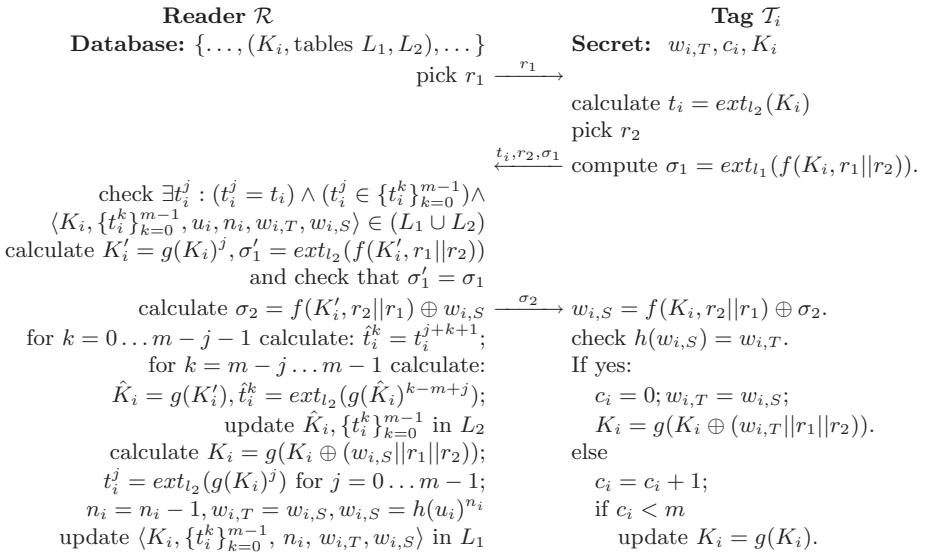


Fig. 1. The backward and forward untraceable RFID protocol

Our attack works nevertheless, as follows, using the basic principle where we intentionally desynchronize the tag from the reader by sending the tag into the future [19].

1. **Learning:** An adversary sends m number of queries r_1^j for $1 \leq j \leq m$ to the tag \mathcal{T}_0 , and records the tag's response t_j for $1 \leq j \leq m$. Since the adversary is impersonating the reader, thus each time it will not pass the check by the tag, and so each time the tag would update its stored secret as $K_i = g(K_i)$, from which t_i will be derived in the next session.
2. **Challenge:** Query r_1^m to the tag $\mathcal{T}_b \in \{\mathcal{T}_0, \mathcal{T}_1\}$, and obtain its response t^* .
3. **Guess:** Check if $t^* = t_m$. If so, then the adversary knows this was the tag it queried during the learning phase i.e. $\mathcal{T}_b = \mathcal{T}_0$. Else, it knows that $\mathcal{T}_b = \mathcal{T}_1$.

It was remarked in [19] that once a tag is successfully authenticated by a reader, then the tag's stored secret K_i would be freshly randomized so that tracing of any kind is prevented. Yet, our adversary can repeat the above step of the Learning phase by sending m arbitrary queries r_1^j for $1 \leq j \leq m$ to the tag again to desynchronize it and the same tracing attack applies.

In order to solve the DoS problem, the authors included into the design a feature that unfortunately allowed our attack causing the tag to be traceable even without corruption, although the goal for their protocol was much stronger i.e. backward and forward untraceability even with corruption.

Violating the Forward Untraceability. Another goal of the protocol is to achieve forward untraceability, i.e. even if a tag is corrupted thus leaking its stored secret K_i , it should be impossible for the adversary to trace the tag in

future sessions. Nevertheless, an attack by the adversary proceeds as follows, using the example application in [19] of a tag embedded in a purchased item: Initially, the seller's reader \mathcal{R}_1 has legitimate access to the tag. At the point of purchase, ownership of this access should transfer to the buyer's reader \mathcal{R}_2 . The attack can be mounted either by the seller's reader or by an outsider adversary having access to `Corrupt` queries.

1. An outsider adversary issues a `Corrupt` query to the tag \mathcal{T}_b , obtaining its stored secret K_i . Alternatively, the seller's reader \mathcal{R}_1 knows the stored secret K_i and $w_{i,T}$.
2. At the point of purchase, the buyer's reader \mathcal{R}_2 interacts with the tag in a protocol session, thus updating K_i . During this time, the adversary eavesdrops on the values r_1, r_2 communicated in the session.
3. Right after the interaction between the tag and the buyer's reader \mathcal{R}_2 , the adversary initiates a protocol session with the tag. Since it knows the previous K_i , and also the latest values of r_1, r_2 , it can recompute the latest $K_i = g(K_i \oplus (w_{i,T} || r_1 || r_2))$ and thus pass the check by the tag without any problem. It can therefore trace the tag in all future sessions, and other readers including the buyer's can no longer successfully interact with the tag.

This result counters the protocol's claim that its ownership transfer is perfect. In [19], it was argued that the protocol achieves forward untraceability under the assumption that the adversary cannot eavesdrop on all future legitimate interactions involving the tag and the reader; the above attack works without violating that assumption. [19] furthermore gives a provable security model for forward untraceability in its Appendix, yet their protocol was not rigorously proven under that model, but instead its security was supported with brief arguments.

4 O-FRAP and O-FRAKE

At AsiaCCS '07, Le et al. [18] presented a universally composable [4] privacy model for RFID protocols, and proposed O-FRAP and O-FRAKE. These two protocols are shown in figures 2 and 3 respectively, on which F denotes a pseudo-random function.

4.1 Tracing O-FRAP

O-FRAP is formally proven to be a secure untraceable RFID protocol in the Le-Burmester-deMedeiros model where corruption of tags is allowed, in the sense that the only information revealed to an adversary is if a party is a tag or a reader. Yet we show here how its untraceable privacy can be violated by presenting a tracing attack that is valid even in a weaker privacy model were corruption possibility is not granted to the adversary.

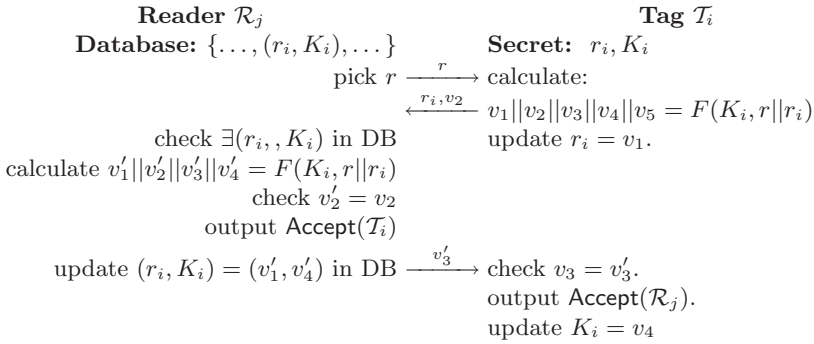


Fig. 2. The O-FRAP protocol

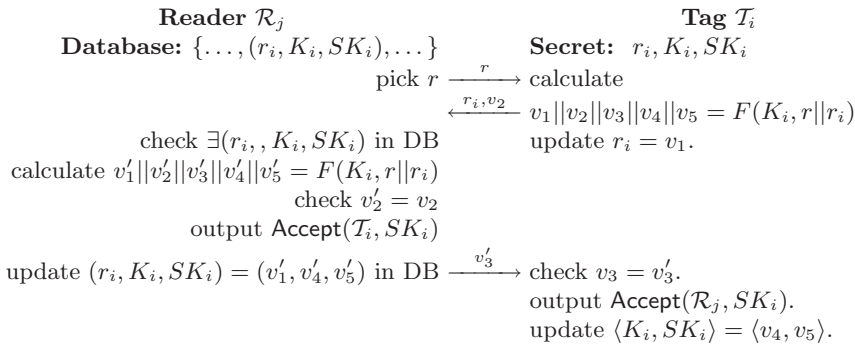


Fig. 3. The O-FRAKE protocol

The attack works as follows:

1. **Learning:** The adversary sends an arbitrary r value to the tag \mathcal{T}_0 , but does not complete the protocol. This causes the tag to update its r_i , while its K_i remains unchanged, thus marking the tag for future tracing.
2. **Challenge:** To trace the tag in future, the adversary observes the interaction between the reader and the tag \mathcal{T}_b .
3. **Guess:** If the reader does not output **Accept**, then the adversary knows that this tag was indeed the tag that it marked in step (1), i.e. $\mathcal{T}_b = \mathcal{T}_0$. Otherwise, he deduces that $\mathcal{T}_b = \mathcal{T}_1$.

4.2 Violating the Forward Privacy of O-FRAP

In the Le-Burmeser-deMedeiros model, corruption is not allowed before a protocol session is initiated, and it is assumed that upon corruption of a party (tag or reader) then the corrupted party's current incomplete session offers no privacy. It is claimed that privacy is maintained for all previously completed sessions involving the heretheto corrupted party.

To motivate our case, we consider the definition of subsession completion in the Le-Burmester-deMedeiros model. A subsession is a party’s view of its current protocol session, e.g. during an O-FRAP protocol session, both the reader and the tag have their own separate views of that session, so-called their subsession. To quote from [18], “Upon successful completion of a subsession, each party accepts its corresponding partner as authenticated.” Thus, at the point where a party outputs `Accept`, its subsession is already considered completed.

Referring to the O-FRAP description in Fig. 2, the reader’s subsession is completed at the point when it outputs `Accept`, i.e. before it updates its entry in L and before it sends v'_3 to the tag. Meanwhile, the tag’s subsession is completed at the point that it outputs `Accept`, i.e. before it updates its K_i . In the context of the Le-Burmester-deMedeiros model, corruption of a party at this point should not violate the privacy of the party corresponding to its completed subsession. This is the problem with the O-FRAP proof that we are exploiting. Indeed, we show how this can be circumvented.

1. The adversary first eavesdrops on an O-FRAP session and records $\langle r, r_i, v_2 \rangle$.
2. It then corrupts a tag T'_i at the point after the tag outputs `Accept`. It thus obtains K'_i corresponding to a previously completed subsession, and not the updated $K'_i = v_4$.
3. The adversary calculates $v_1^* || v_2^* || v_3^* || v_4^* = F(K'_i, r || r_i)$. It can then check the computed v_2^* with its recorded v_2 for a match, thereby associating the tag T'_i to the particular completed subsession corresponding to its recorded $\langle r, r_i, v_2 \rangle$.

Our attack here requires a stronger adversary than the other attacks we have presented in earlier sections of this paper, yet it fits into the Le-Burmester-deMedeiros model for which O-FRAP’s privacy was proven, and shows that O-FRAP does not achieve its goal of forward untraceable privacy.

It appears that O-FRAP can be made to resist this attack by having the tag output `Accept` as the very last step of the protocol, i.e. after K_i has been updated.

4.3 Breaking the Forward Secrecy of O-FRAKE

The above attack can be extended to break the forward secrecy of the O-FRAKE protocol, which is an extension of O-FRAP that furthermore establishes a shared secret session key between the tag and reader.

1. The adversary first eavesdrops on an O-FRAKE session and records $\langle r, r_i, v_2 \rangle$.
2. It then corrupts a tag T'_i at the point after the tag outputs `Accept`. It thus obtains $\langle K'_i, SK'_i \rangle$ corresponding to a previously completed subsession, and not the updated $\langle K'_i, SK'_i \rangle = \langle v_4, v_5 \rangle$.
3. The adversary calculates $v_1^* || v_2^* || v_3^* || v_4^* || v_5^* = F(K'_i, r || r_i)$. It can then check the computed v_2^* with its recorded v_2 for a match, thereby associating the tag T'_i to the particular completed subsession corresponding to its recorded $\langle r, r_i, v_2 \rangle$; and further it also knows that the established session key for that associated session is SK'_i .

Similarly, it appears that O-FRAKE can be made to resist this attack by having the tag output `Accept` as the very last step of the protocol, i.e. after $\langle K_i, SK_i \rangle$ have been updated.

5 Concluding Remarks

We described in an alternative manner the privacy models that capture the notion of untraceable privacy (UPriv) and briefly discussed its relation to existing models. The aim was to use this notion to show how some recent provably secure RFID protocols (with proofs of security in strong adversarial models) do not achieve this privacy notion even under the weak adversarial model that does not require corruption of tags. In some sense, these results support the case [7, 8, 9, 10, 11, 12, 27, 28] that while provable security is the right approach to design and analysis of protocols, more careful analysis and interpretation of provable security models and proofs are needed to ensure the right definitions [30] are put in place.

As a commonly accepted model addressing privacy and security in RFID has to be established and many RFID protocols are proposed without providing any formal security proof, these results strengthen the need for such a model to facilitate better design of RFID protocols that offer both privacy and security.

“Big Brother is watching you”.

George Orwell, *Nineteen Eighty-Four*.

Acknowledgements

We thank the anonymous referees for constructive comments.

References

1. Albertsons Announces Mandate, RFID Journal, 5 March (2004), <http://www.rfidjournal.com/article/articleview/819/1/1/>
2. Avoine, G.: Adversarial Model for Radio Frequency Identification. Cryptology ePrint Archive, report 2005/049, 20, Available at IACR ePrint Archive (February 2005), <http://eprint.iacr.org/2005/049>
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: Proc. IEEE FOCS 2001, pp. 136–145 (2001) Full version available at IACR ePrint Archive (last revised, 13 December 2005), <http://eprint.iacr.org/2000/067>
5. Carluccio, D., Lemke, K., Paar, C.: E-Passport: The Global Traceability or How to Feel Like a UPS Package. In: Lee, J.K., Yi, O., Yung, M. (eds.) WISA 2006. LNCS, vol. 4298, pp. 391–404. Springer, Heidelberg (2007)

6. CASPIAN, Boycott Benetton (accessed 19, September 2007), <http://www.boycottbenetton.com>
7. Choo, K.-K.R.: Refuting Security Proofs for Tripartite Key Exchange with Model Checker in Planning Problem Setting. In: Proceedings of IEEE CSFW 2006, pp. 297–308 (2006)
8. Choo, K.-K.R., Hitchcock, Y.: Security Requirements for Key Establishment Proof Models: Revisiting Bellare-Rogaway and Jeong-Katz-Lee Protocols. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 429–442. Springer, Heidelberg (2005)
9. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: On Session Key Construction in Provably-Secure Key Establishment Protocols. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 116–131. Springer, Heidelberg (2005)
10. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining Indistinguishability-based Proof Models for Key Establishment Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
11. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Errors in Computational Complexity Proofs for Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)
12. Choo, K.-K.R., Boyd, C., Hitchcock, Y., Maitland, G.: On Session Identifiers in Provably Secure Protocols. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 351–366. Springer, Heidelberg (2005)
13. Heydt-Benjamin, T.S., Bailey, D.V., Fu, K., Juels, A., O'Hare, T.: Vulnerabilities in First-Generation RFID-enabled Credit Cards. In: Proceedings of Financial Cryptography 2007. LNCS, vol. 4886, pp. 2–14. Springer, Heidelberg (2008)
14. Hoepman, J.-H., Hubbers, E., Jacobs, B., Oostdijk, M., Schreur, R.W.: Crossing Borders: Security and Privacy Issues of the European e-Passport. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 152–167. Springer, Heidelberg (2006)
15. Juels, A., Molnar, D., Wagner, D.: Security and Privacy Issues in E-Passports. In: Proceedings of SecureComm 2005, pp. 74–88 (2007) Full version available at IACR ePrint Archive (last revised 18 September 2007), <http://eprint.iacr.org/2005/095>
16. Juels, A., Weis, S.A.: Defining Strong Privacy for RFID. In: Proceedings of PerCom 2007, April 7, 2006, pp. 342–347 (2007) Full version available at IACR ePrint Archive <http://eprint.iacr.org/2006/137>
17. Kosta, E., Meints, M., Hensen, M., Gasson, M.: An Analysis of Security and Privacy Issues Relating to RFID Enabled ePassports. In: Proceedings of IFIP SEC 2007, IFIP 232, pp. 467–472 (2007)
18. Le, T.V., Burmester, M., de Medeiros, B.: Universally Composable and Forward-Secure RFID Authentication and Authenticated Key Exchange. In: Proceedings of ASIACCS 2007, February 14, 2007, pp. 242–252 (2007); Full version titled Forward-Secure RFID Authentication and Key Exchange, IACR ePrint Archive, <http://eprint.iacr.org/2007/051>,
19. Lim, C.H., Kwon, T.: Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 1–20. Springer, Heidelberg (2006)
20. Michelin Embeds RFID Tags in Tires, RFID Journal (January 17, 2003), <http://www.rfidjournal.com/article/articleview/269/1/1/>
21. Mitsubishi Electric Asia Switches on RFID, RFID Journal (September 11, 2006), <http://www.rfidjournal.com/article/articleview/2644/>

22. Monnerat, J., Vaudenay, S., Vuagnoux, M.: About Machine-Readable Travel Documents: Privacy Enhancement using (Weakly) Non-Transferable Data Authentication. In: Proceedings of RFIDSec 2007, pp. 15–28 (2007)
23. Naor, M., Yung, M.: Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: Proceedings of STOC 1990, pp. 427–437 (1990)
24. Ohkubo, M., Suzuki, K., Kinoshita, S.: RFID Privacy Issues and Technical Challenges. *Communications of the ACM* 48(9), 66–71 (2005)
25. Ouafi, K., Phan, R.C.-W.: Privacy of Recent RFID Authentication Protocols. In: Proceedings of ISPEC 2008. LNCS, vol. 4991, pp. 263–277. Springer, Heidelberg (2008)
26. Paise, R.I., Vaudenay, S.: Mutual Authentication in RFID. In: Proceedings of AsiaCCS 2008 (to appear, 2008)
27. Phan, R.C.-W., Goi, B.-M.: Cryptanalysis of the N-Party Encrypted Diffie-Hellman Key Exchange using Different Passwords. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 226–238. Springer, Heidelberg (2006)
28. Phan, R.C.-W., Goi, B.-M.: Cryptanalysis of Two Provably Secure Cross-Realm C2C-PAKE Protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 104–117. Springer, Heidelberg (2006)
29. Rackoff, C., Simon, D.R.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 434–444. Springer, Heidelberg (1992)
30. Rogaway, P.: On the Role Definitions in and Beyond Cryptography. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 13–32. Springer, Heidelberg (2004)
31. Schnorr, C.P.: Efficient signature generation by smart cards. *Journal of Cryptology* 4(3), 161–174 (1991)
32. Target, Wal-Mart Share EPC Data, *RFID Journal* (October 17, 2005), <http://www.rfidjournal.com/article/articleview/642/1/1/>
33. Tsudik, G.: YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In: Proceedings of PerCom 2006, pp. 640–643 (2006)
34. Vaudenay, S.: RFID Privacy based on Public-Key Cryptography. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 1–6. Springer, Heidelberg (2006)
35. Vaudenay, S.: On Privacy Models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)

The Security of EPC Gen2 Compliant RFID Protocols

Mike Burmester¹ and Breno de Medeiros²

¹ Department of Computer Science,
Florida State University, Tallahassee, FL 32306, USA
burmester@cs.fsu.edu

² Information Security Consultant,
(Recently with Google Inc.)
Santa Clara, CA 95054, USA
breno@brenodemedeiros.com

Abstract. The increased functionality of EPC Class1 Gen2 (EPCGen2) is making this standard the de facto specification for inexpensive tags in the RFID industry. EPCGen2 supports only very basic security tools such as a 16-bit Pseudo-Random Number Generator and a 16-bit Cyclic Redundancy Code. Recently two EPCGen2 compliant protocols that address security issues were proposed in the literature. In this paper we analyze these protocols and show that they are not secure and subject to replay/impersonation and synchronization attacks. We then consider the general issue of supporting security in EPCGen2 compliant protocols and propose two RFID protocols that are secure within the restricted constraints of this standard, and an anonymous RFID mutual authentication protocol with forward secrecy that is compliant with the EPC Class2 Gen2 standard.

Keywords: EPCGen2 compliance, security, anonymity, forward secrecy, unlinkability.

1 Introduction

Radio Frequency Identification (RFID) is a promising new technology that is envisioned to replace barcodes and to be massively deployed for inventory management, supply-chain logistics and retail operations. The advantage of RFID over barcode technology is that it is wireless and does not require direct line-of-sight reading. Furthermore, RFID readers can interrogate tags at greater distances, much faster and concurrently. Perhaps one of the most important advantages of RFID technology is that tags have read/write capability, allowing stored tag information to be altered dynamically. A typical RFID system has three components: tags, one or more readers, and a backend server. The communication channel between the reader and the backend server is (usually) assumed to be secure while the (wireless) channel between the reader and the tag is insecure.

To foster and promote the adoption of RFID technology and to support interoperability, EPCGlobal [13] and the International Organization for Standards

(ISO) [16] have been actively engaged in defining standards for tags, readers, and the communication protocols between them. A recently ratified standard is EPC Class 1 Gen 2 (EPCGen2). This is a communication standard that creates a platform on which to build interoperable RFID protocols. It supports efficient tag reading, flexible bandwidth use, multiple read/write capabilities and basic security guarantees, provided by an on-chip 16-bit Pseudo-Random Number Generator (RNG) and a 16-bit Cyclic Redundancy Code (CRC-16). EPCGen2 is designed to strike a balance between cost and functionality, with less attention paid to security which, arguably, at this stage in the development of RFID technology may be justified.

In this paper we are concerned with the security of EPCGen2 compliant protocols. We recognize that there are situations in which one has to design security into systems with restricted capability so as to promote low-cost widespread use. In such situations it is important to employ protocols that offer the best level of security within the constraints of the specification. Several RFID authentication protocols that address security issues using lightweight cryptographic mechanisms have been proposed in the literature. Most of these use hash functions [21,20,25,15,13,11] and [23,12,19], which are beyond the capability of most low-cost tags and are not supported by EPCGen2. Some protocols use pseudo-random number generators [25,17,5,4,24], a mechanism that is supported by EPCGen2, but these are not optimized for EPCGen2 compliance. Other protocols use timestamps (*e.g.* [23]), however these are also not supported by EPCGen2.

The research literature for RFID security is already quite extensive and growing. We refrain from a comprehensive review of the literature, and refer the interested reader to a fairly comprehensive repository available online at [2]. Recently two RFID authentication protocols specifically designed for compliance with EPCGen2 have been proposed [10,9]. These combine the CRC-16 of the EPCGen2 standard with its 16-bit RNG to hash, randomize and link protocol flows, and to prevent cloning, impersonation and denial of service attacks. In this paper we analyze these protocols and show that they do not achieve their security goals. One may argue that, because the security of EPCGen2 is set to only 16-bits, any RFID protocol is potentially vulnerable, for example to ciphertext-only attacks that exhaust the 16-bit range of the components of protocol flows. While this is certainly the case, such attacks may be checked by using additional keying material and by constraining the application (*e.g.*, the life-time of tags). We contend that there is scope for securing low cost devices. Obviously, the level of security may not be sufficient for sensitive applications. However there are many low cost applications where there is no alternative.

The rest of this paper is organized as follows. Section 2 introduces the EPCGen2 standard focusing on security issues. Section 3 analyzes two recently proposed EPCGen2 protocols. Section 4 describes two “trivial” RFID protocols whose security is reduced to the security constraints of EPCGen2, and an anonymous RFID protocol that complies with the EPC Class 2 Gen2 standard. Section 5 considers an extension that captures a kill functionality.

2 The EPCGen2 Standard

The EPC Class 1 Generation 2 UHF Air Interface Protocol [13] specifies the operation and functionality of passive RFIDs. It defines the physical and logical requirements for interrogator-talks-first systems that operate in the 860-960 MHz frequency range. Readers (interrogators) transmit information to tags by modulating an RF signal. Tags receive both transmitted information and operating energy from the RF signal. The readers receive information from the tags by transmitting a continuous-wave RF signal, which the tags modulate (backscatter). EPCGen2 deals with the medium access control layer (air interface) and the tag identification layer (physical interactions) of RFID systems. These layers involve RF signaling, managing tag populations, tag singulation (identifying individual tags), collision arbitration (resolving collisions in multi-tag environments), and conformance requirements. A particular attractive feature of EPCGen2 is that it provides for high-speed reading and sortation. The following minimal on-chip tag persistent memory (non-volatile) features are specified:

- Reserved memory that contains a 32-bit kill password (KP) to permanently disable the tag and a 32-bit access password (AP).
- EPC memory that contains: the parameters of a cyclic redundancy code CRC-16 (16 bits), protocol control (PC) bits (16 bits), and an electronic product code EPC that identifies the object to which the tag is (or will be) attached (at least 32 bits).
- TID memory that contains sufficient information to identify to a reader the (custom/optional) features that a tag supports and tag/vendor specific data.
- User memory that allows user-specific data storage.

EPCGen2 also provides for optional user memory and password-protected access control. Two basic mechanisms to protect the tag \leftrightarrow reader channels are supported:

- A 16-bit Pseudo-Random Number Generator and,
- A 16-bit Cyclic Redundancy Code.

2.1 The Pseudo-random Number Generator

A pseudo-random number generator (RNG) is a deterministic function that on input a random binary string, called *seed*, outputs a sequence of numbers that are indistinguishable from random numbers. RNGs are constructed from pseudo-random bit generators (RBGs). A RBG stretches a random seed of length k to a pseudo-random binary string of length $\ell \gg k$. This string can be partitioned into blocks of length m to get an m -bit RNG. The length of the random seed must be selected carefully to guarantee that the numbers generated are pseudo-random.

Each number generated by a RNG is said to be *drawn* by the generator. If the numbers drawn by a RNG (or the bits drawn by a RBG) cannot be predicted given the outcomes of prior draws, then the RNG (RBG) is said to be cryptographically secure.

EPCGen2 does not detail the structure of the implemented 16-bit RNG; however it does specify the minimum security levels that should be supported:

1. **Probability of RN16:** The probability that a pseudo-random number RN16 drawn from the RNG has value r is bounded by:

$$0.8/2^{16} < Prob(RN16 = r) < 1.25/2^{16}.$$

2. **Drawing identical sequences:** For a tag population of up to 10,000 tags, the probability that any two or more tags simultaneously draw the same sequence of RN16s is $< 0.1\%$, regardless of when the tags are energized.
3. **Next-number prediction:** A RN16 drawn from a tag’s RNG is not predictable with probability better than 0.025% , given the outcomes of all prior draws.

Strength of EPCGen2 Compliant RNGs. The strength of a cryptographic RNG is usually expressed in terms of the average maximum length of the sequences of drawn numbers that are indistinguishable from random sequences. With this in mind, we interpret the requirements above as imposing minimal security requirements on the RNGs specified by EPCGen2:

1. The first “randomness” requirement implies that the value of a drawn number cannot be too biased. This requirement, while certainly satisfied by cryptographically secure RNGs, is actually too weak *by itself* to imply much in the way of pseudo-randomness. For instance, a simple counter that increments from 0 to $2^{16} - 1$ and then cycles back satisfies this condition.
2. The second “collision” requirement needs a detailed examination. A well known approximation for the collision problem is $n = \sqrt{2d \ln(1/(1-p))}$ [22,18], where n is the number of integers drawn randomly with uniform distribution, d the size of the range of the integers, and p the probability that at least two integers have the same value. From this approximation we see that when n is close to \sqrt{d} then the probability of collision is greater than 50%. When $p = 0.1\%$ and $n = 10,000$ we get that d is approximately 2^{36} . That implies that observing two successive numbers (36-bits) drawn from the RNG of two or more tags still guarantees a fairly unbiased (and random-looking) sequence, at least as far as collisions are concerned.
3. The third EPC requirement for the next-number prediction also needs examination. Let RBG be the pseudo-random bit generator that defines the RNG. For the cryptographic security of RBG, the next-bit prediction should be $p = 0.5 + \varepsilon$, ε negligible. This expression can be used to compute the next-number prediction. Let B_0 be the bit-sequence of prior numbers. The prediction for the next 16-bit number is:

$$P = \prod_{i=0}^{i=15} Prob(b_{i+1}|B_0b_0 \cdots b_i) = p^{16} = (0.5 + \varepsilon)^{16},$$

where $b_1, \dots, b_i, b_{i+1}, \dots$, are the bits drawn by the RBG after B_0 . EPCGen2 bounds P by 0.025% . Taking $(0.5 + \varepsilon)^{16} < 0.025\%$ we see that ε is bounded 0.094 . This is not sufficiently small to provide cryptographic security for the 16-bit RNG, and suggests that the 0.025% bound should be lowered at least one order.

In conclusion, in particular with respect to Condition 2, a reasonably conservative assumption is that the standard for EPCGen2 guarantees the pseudo-randomness of at least two RN16s (32-bits), and probably more, before there is a need to re-seed the key. If Condition 3 were to be interpreted in the absence of restraints on the number of prior draws, then at least 3-4 RN16s (48-64-bits) drawn from the RNG should be indistinguishable from pseudo-random.

2.2 The 16-bit Cyclic Redundancy Code

CRCs are error-detecting codes that check accidental (non-malicious) errors caused by faults during transmission. In EPCGen2, a CRC-16 is used to protect the information transmitted by both readers and tags. The CRC-16 algorithm maps arbitrary length inputs onto 16-bit outputs as follows: an n -bit input p is first replaced by a binary polynomial $p(x)$ of degree $n - 1$, and then reduced modulo a specific polynomial $g(x)$ of degree 16 to a polynomial remainder $r(x) : p(x) = q(x)g(x) + r(x)$. The remainder has degree less than 16 and corresponds to a 16-bit number. For EPCGen2, the polynomial $g(x)$ is the irreducible polynomial: $x^{16} + x^{12} + x^5 + 1$ (over the finite field $GF(2)$ of two elements). CRC-16 will detect burst errors of 16-bits or less, any odd number of errors less than 16, and error patterns of length 2 [13].

CRCs by themselves are not suitable for protecting against intentional (malicious) alteration of data. They do not provide the one-wayness required by message digest codes: they are linear codes whose one-wayness is comparable to xor-sums.

3 Weaknesses of Currently Proposed EPCGen2 Compliant RFID Protocols

In this section we consider two recently proposed EPCGen2 compliant protocols: the Duc-Park-Lee-Kim protocol [10] and the Chien-Chen protocol [9]. The first protocol is designed to support untraceability and uncloneability; the second to support the same security features, but also to provide forward secrecy. We shall show that both protocols fall short of their claimed security.

In the protocols below we use the following notation: \mathcal{S} is the backend server, \mathcal{R} the reader, \mathcal{T} the tag. We assume that \mathcal{S} and \mathcal{R} are linked with a secure channel, and for simplicity, only consider the case when the authentication is online.

3.1 The Duc-Park-Lee-Kim RFID Protocol and Its Weaknesses

In this protocol [10] each tag \mathcal{T} shares two values with the backend server \mathcal{S} : a 32-bit key and a 16-bit key. The tag stores these in its non-volatile memory and the server \mathcal{S} stores them in a database DB . The 16-bit key is initialized

$K \leftarrow K_{init}$ and updated with each successful authentication of the tag by the server. The 32-bit key is assigned the tag’s EPC. The protocol has four passes.

1. $\mathcal{R} \rightarrow \mathcal{T}$: a query request.
2. $\mathcal{T} \rightarrow \mathcal{R} \rightarrow \mathcal{S}$: a random 16-bit nonce r , $M_1 = CRC(EPC||r) \oplus K$, and $C = CRC(r \oplus M_1)$.
 \mathcal{S} checks C and that: $M_1 \oplus K = CRC(EPC||r)$ for some (K, EPC) in DB .
 If the checksum fails or if it cannot find a match then it rejects \mathcal{T} .
 \mathcal{S} computes $M_2 = CRC(EPC||AP||r) \oplus K$ and updates the key K of \mathcal{T} in DB : $K \leftarrow RNG(K)$.
3. $\mathcal{S} \rightarrow \mathcal{R}$: M_2 , and details that identify the object to which the tag is attached, depending on the reader’s privileges.
4. $\mathcal{R} \rightarrow \mathcal{T}$: M_2 , “end session”.
 \mathcal{T} checks that: $M_2 \oplus K = CRC(EPC||AP||r)$. If this is valid then it updates the key K : $K \leftarrow RNG(K)$.

This protocol is subject to a synchronization attack as observed by Chien-Chen in [9]: if the adversary prevents the tag in Pass 4 from receiving an “end session” instruction, the tag will not update its key while the server will have updated the corresponding key in DB . Consequently the server will not be synchronized with the tag and any future attempts by the tag to get authenticated will fail. However there is another important weakness, caused by the linearity of CRC-16. The adversary can easily forge the response (r', M'_1, C') of a tag \mathcal{T} in any session by simply using an earlier response (r, M_1, C) obtained by interrogating \mathcal{T} (as a rogue reader) or eavesdropping. Indeed let:

1. r' be a random 16-bit number.
2. $A = CRC(00||r \oplus r')$ and $B = CRC(A \oplus r \oplus r')$.
3. $M'_1 = M_1 \oplus A = [CRC(EPC||r) \oplus K] \oplus CRC(00||r \oplus r')$
 $= CRC(EPC||r') \oplus K$.
4. $C' = C \oplus B = CRC(M_1 \oplus r) \oplus CRC(A \oplus r \oplus r') = CRC(M_1 \oplus A \oplus r')$
 $= CRC(M'_1 \oplus r')$.

Clearly (r', M'_1, C') is valid for *any* query request, and so the adversary will succeed in impersonating the tag \mathcal{T} .

3.2 The Chien-Chen RFID Protocol and Its Weaknesses

This protocol [9] is an extension of the Duc-Park-Lee-Kim protocol, designed to address its weaknesses, as well as to offer forward secrecy. Each tag \mathcal{T} stores three values in non volatile memory: a 32-bit EPC, a 16-bit key K and a 16-bit access key P . For each tag the backend server \mathcal{S} stores six values in a database DB : the tag’s EPC, two 16-bit keys K_{old}, K_{new} , two 16-bit access keys P_{old}, P_{new} and $DATA$. The values of the key K and the access key P that the tag stores and the corresponding values of the keys K_{old}, K_{new} and P_{old}, P_{new} that the server stores are updated with each successful tag authentication so as to preserve synchronization. The protocol is described below.

1. $\mathcal{R} \rightarrow \mathcal{T}$: a random nonce N_1 .
2. $\mathcal{T} \rightarrow \mathcal{R} \rightarrow \mathcal{S}$: a random nonce N_2 , and $M_1 = CRC(EPC||N_1||N_2) \oplus K$.
 \mathcal{S} checks that $M_1 = CRC(EPC||N_1||N_2) \oplus K_j$ for some (K_j, EPC) , $j \in \{new, old\}$, in DB .
 If \mathcal{S} cannot find a match then it rejects \mathcal{T} and sends a “failure” message to \mathcal{R} .
 Else it updates the keys of \mathcal{T} : $K_{old} \leftarrow K_{new} \leftarrow RNG(K_{new})$, $P_{old} \leftarrow P_{new} \leftarrow RNG(P_{new})$ in DB , and computes $M_2 = CRC(EPC||N_2) \oplus P_j$, $j \in \{new, old\}$, using the j -value in M_1 .
3. $\mathcal{S} \rightarrow \mathcal{R}$: M_2 and $DATA$, with product information.
4. $\mathcal{R} \rightarrow \mathcal{T}$: M_2 .
 \mathcal{T} checks that $M_2 = CRC(EPC||N_2) \oplus P$.
 If this is valid it updates its keys: $K \leftarrow RNG(K)$, $P \leftarrow RNG(P)$.

There are several weaknesses with this protocol. One weakness concerns the synchronization of keys in Pass 4: the protocol does not protect tags against repeated synchronization attacks. Indeed, the first time the adversary prevents the tag from getting its confirmation in Pass 4, the tag will not update (K, P) , while the server will have updated the corresponding values of the tag in DB : $K_{old} \leftarrow K_{new}$, $K_{new} \leftarrow RNG(K_{new})$ and $P_{old} \leftarrow P_{new}$, $P_{new} \leftarrow RNG(P_{new})$. Then, if the value of (K, P) stored by the tag prior to the attack was (K_{new}, P_{new}) , after the attack it will be (K_{old}, P_{old}) . Suppose the attack is repeated. The server will accept the tag’s response in Pass 2 because the tag uses the value (K_{old}, P_{old}) in DB . But this will be discarded by the server when it updates its keys. However the tag will not update its keys in Pass 4 if it is prevented from getting a confirmation M_2 . It follows that the adversary will succeed in desynchronizing the tag after the second attempt. Desynchronization will also result from two successive reading failures by a tag.

This protocol shares the weakness of the Duc-Park-Lee-Kim protocol resulting from the linearity of CRC-16. In this case, the adversary can forge a response N'_2, M'_1 to any challenge N'_1 of the server by using an earlier protocol flow $(N_1; N_2, M_1)$ obtained by interrogating the tag (as a rogue reader) or eavesdropping. Indeed, let:

1. N'_2 be a random nonce.
2. $B_1 = N'_1 \oplus N_1$ and $B_2 = N'_2 \oplus N_2$.
3. $A = CRC(00||B_1||B_2)$.
4. $M'_1 = M_1 \oplus A = [CRC(EPC||N_1||N_2) \oplus K] \oplus [CRC(00||B_1||B_2)]$
 $= CRC(EPC||N'_1||N'_2) \oplus K$.

Then N'_2, M'_1 is a valid response to the challenge N'_1 .

4 Secure EPCGen2 Protocols

We next consider three “trivial” RFID authentication protocols (TRAPs) that comply with EPCGen2, and whose security is reduced to the minimum levels of statistical behavior of RNGs guaranteed by this standard.

The first protocol TRAP-0, is a toy example to illustrate that the RNG of EPCGen2 cannot be used to securely link protocol flows. We then show how to modify the RNG so that we get security. This will give us TRAP-1. The next protocol, TRAP-2, is an extension that provides anonymity, but not forward secrecy. The last protocol, TRAP-3, supports strong privacy (with forward secrecy).

4.1 A Trivial RFID Protocol That is EPCGen2 Compliant

This protocol uses the 16-bit RNG supported by EPCGen2, seeded with a 16-bit key K . For each tag \mathcal{T} with identifier $id(\mathcal{T})$, the server \mathcal{S} stores in a database DB an entry of the form: $\langle id(\mathcal{T}), K \rangle$, used to identify the tag. We assume that the server \mathcal{S} and the reader \mathcal{R} are linked by a secure (private and authenticated) channel.

TRAP-0

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N .
 \mathcal{T} computes $L = K \oplus N$ and draws M from $RNG(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: $id(\mathcal{T}), M$.
 \mathcal{S} computes $L' = K \oplus N$, where K is the key of $id(\mathcal{T})$, and draws M' from $RNG(L')$.
 If $M' = M$ then the tag \mathcal{T} is authentic. Else it is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: “end session”.

4.2 Analysis of TRAP-0

The security of TRAP-0 is based on the statistical behavior of the RNG of EPCGen2 as specified by the three EPCGen2 constraints in Section 2.1. TRAP-0 has two major weaknesses that result from the fact that a 16-bit RNG is used as a security tool to link the challenge-response flows of a protocol instance. The first concerns *exhaustive-key* attacks: since RNGs are deterministic, an exhaustive search on all possible 2^{16} key (seed) values can be used to determine the key (alternatively, a pre-computed table of RNG entries can be used). One way to prevent such attacks is to use additional keying material (we shall do this in Section 4.4).

A second weakness concerns *related-key* attacks: EPCGen2 does not specify any protection of its RNGs against attacks in which the adversary exploits values drawn from RNGs whose keys are related. We next discuss these attacks, and consider an approach that may be used to deal with them.

The related-key problem

- *Search problem.* Given $RNG(K \oplus N_i)$, N_i , $i = 1, \dots, t$ and $N \neq N_i$: find $RNG(K \oplus N)$.
- *Decision problem.* Given $RNG(K \oplus N_i)$, N_i , $i = 1, \dots, t$ and $N, X \neq N_i$: is $X = RNG(K \oplus N)$?

Clearly if the adversary can compute $RNG(K \oplus N)$ for a fresh nonce N , given a history of $RNG(K \oplus N_i)$, N_i , $i = 1, \dots, t$, obtained by eavesdropping on protocol flows, then the adversary can forge a session with challenge N . Although we have not as yet presented an anonymous TRAP (we shall do this in Section 4.5), if values drawn from a RNG are used as pseudonyms, then the adversary will be able to disambiguate tags if it can solve the related-key decision problem. The related-key problem described above is for passive attacks. In an active attack, the adversary can select the numbers N_i , $i = 1, \dots, t$, adaptively (but not N or X). Protecting protocols against adaptive attacks is usually much harder.

There are several ways to deal with such attacks. The solution we consider here involves modifying the 16-bit EPCGen2 RNG so that it is hard to link its inputs and outputs.

4.3 Constructing a PRF Family from a RNG

We briefly describe a construction, due to Goldreich-Goldwasser-Micali [14]. Let G be an n -bit RNG and K an n -bit number. Denote by $G_0(K)$ the first n -bit number output by G and $G_1(K)$ the next n -bit number. Let $X = X_1, X_2, \dots, X_t$, $t \geq n$, be a t -bit number and $G_X(K) = G_{X_t}(Z_{t-1})$, where

$$Z_{t-1} = (G_{X_{t-1}}(\dots(G_{X_1}(G_{X_0}(K)))) \dots).$$

Define the function $f_K : \{0, 1\}^t \rightarrow \{0, 1\}^n$ by $f_K(X) = G_X(K)$. It is shown in [14] that the family $F_n = \{f_K\}_{|K|=n}$ is a pseudo-random function (PRF).

In our TRAP protocols we shall use $f_K(X)$ as a RNG: since F_n is a PRF, $f_K(X)$ is secure against attacks that exploit correlated values of X . The first number drawn from $f_K(X)$ will be $G_X(K)$. If a second number has to be drawn then, in the last step of the construction above we take either the second or the third n -bit number output by $G(Z_{t-1})$, depending on whether $X_t = 0$ or $X_t = 1$; for the t -th draw, we take either the t -th output number or the $(t+1)$ -th output number of $G(Z_{t-1})$.

4.4 TRAP-1: A Trivial EPCGen2 RFID Protocol

To deal with exhaustive key attacks and related-key attacks on TRAP-0 we use two 16-bit numbers K_0, K_1 as key, and evaluate $f_{K_0}(K_1 \oplus \cdot)$ on 16-bit number inputs. In particular, for the 16-bit number N , we draw numbers from $f_{K_0}(K_1 \oplus N)$ instead of $f_K(N)$. Observe that if N and $L = f_{K_0}(K_1 \oplus N)$ are given, then there are roughly 2^{16} pairs (K'_0, K'_1) for which $L = f_{K'_0}(K'_1 \oplus N)$, each one being equally likely to be (K_0, K_1) . The search range is narrowed as more values (N_i, L_i) become available: to prevent attacks in which partial information about the seed may be leaked, the tag's key (K_0, K_1) may be updated during the execution of the protocol. We shall do this in our last TRAP protocol (Section 4.7). Finally note that since the value $f_{K_0}(K_1 \oplus N)$ is the output of an EPCGen2 RNG, it is subject to the minimal security requirements of EPCGen2 given in Section 2.

TRAP-1

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N .
 \mathcal{T} computes $L = K_1 \oplus N$, and draws a 16-bit number M from $f_{K_0}(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: $id(\mathcal{T})$, M .
 \mathcal{S} computes $L' = K_1 \oplus N$, where $K = K_0 || K_1$ is the key of $id(\mathcal{T})$, and draws M' from $f_{K_0}(L')$.
 If $M' = M$ then the tag \mathcal{T} is authentic. Else it is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: Details of \mathcal{T} (obtained from TID), “end session”.

Security is based on the fact that $f_{K_0}(K_1 \oplus \cdot)$ is a PRF, since it is generated by a RNG. The numbers drawn from $f_{K_0}(K_1 \oplus N)$ are 16-bit numbers drawn from an EPCGen2 RNG, pseudo-randomly rearranged to thwart related-key attacks. These numbers are bound by the EPC constraints in Section 2.1. Consequently we have:

1. *Robustness against passive attacks.* Suppose that the adversary obtains the values of authenticators M of one or more tags for several sessions. Since these are drawn using independent seeds, the probability of predicting the next authenticator is bounded by the EPCGen2 probability of drawing a number from a RNG.
2. *Robustness against active attacks.* Since the tag’s response is linked to the reader’s challenge, a replay attack will fail. Furthermore, since the numbers M generated by a tag \mathcal{T} are drawn from its RNG (with key K_0), by the next-number EPCGen2 prediction requirement, the adversary cannot predict their value with probability better than 0.025%.

The security of the TRAP protocols is discussed in a more formal setting in Section 4.6 and the Appendix.

4.5 TRAP-2: A Trivial Anonymous EPCGen2 RFID Protocol

Our next protocol, TRAP-2, extends TRAP-1 to capture anonymity. For this protocol each tag \mathcal{T} stores two numbers, a 16-bit pseudonym P and a 32-bit key $K = K_0 || K_1$; the server \mathcal{S} stores in a database DB for each tag an entry of the form: $\langle id(\mathcal{T}), K \rangle$. Initially P and K are assigned random values.

TRAP-2

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N .
 \mathcal{T} computes $L = (K_1 \oplus P) || N$ and draws two 16-bit numbers M, M_1 from $f_{K_0}(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M .
 \mathcal{T} updates $P \leftarrow M_1$.
 \mathcal{S} computes $L' = (K_1 \oplus P) || N$ and draws M' from $f_{K_0}(L')$ for every key $K = K_0 || K_1$ in DB . If there is a match $M' = M$ then tag \mathcal{T} is authentic. Else it is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: Details regarding tag \mathcal{T} , and “end session”.

TRAP-2 is a simple extension of TRAP-1. Anonymity is assured because the pseudonyms P are pseudo-random, and because they are updated after each interrogation (by authorized or rogue readers). We shall discuss in more detail the security of this, and the other TRAP protocols, in the following section.

4.6 The Security of EPCGen2 Compliant Protocols

EPCGen2 is a standard for Class 1 tags that have restricted memory (particularly non-volatile) and circuit footprint. It provides for high speed reading and sortation of tags. This standard focuses on reliability and efficiency and will only support a very basic level of security, provided by a 16-bit RNG. In our first two protocols, TRAP-1 and TRAP-2 we have used this RNG to secure protocol flows: we added an extra 16-bits of keying material to make it harder to invert the RNG and to prevent related-key attacks. Obviously this will affect the efficiency of the interrogating process: identification will take longer and may lead to reading failures. Also the additional non-volatile memory means that the TRAP protocols can only be used with tags that are at the top end of the EPCGen2 standard.

The EPC Air Interface Protocol [13] covers a wide range of tag types. The Class 1 tags we shall consider in this paper are the most basic passive tags. Class 2 tags are covered by the same specifications, but are allowed to have additional memory. In the last part of this section we consider TRAP-3, a Class 2 mutual authentication RFID protocol that supports strong privacy (with forward secrecy). For this protocol we require that each tag has a 48-bit key, and use a 32-bit RNG. We shall assume the same minimum security level of EPCGen2 as specified in Section 2.1, extended to allow for 32-bit RNGs.

Our TRAP protocols are based on the protocols O-TRAP [5] and O-FRAP [24] that are secure in the Universal Composability (UC) framework [6,7,8]. The main feature of these protocols is that their protocol flows are pseudo-random. In the Appendix we describe O-TRAP and show how to adapt it to get a security proof for TRAP-2 in the UC framework, provided a sufficiently large seed for the RNG is used, so as to prevent the adversary from distinguishing its output from random numbers. Similarly, the security proof of O-FRAP can be adapted to get a proof for TRAP-3.

4.7 An EPC Class2 Gen2 Compliant Protocol That Supports Strong Privacy

Our last protocol is a mutual authentication protocol. This protocol uses a 32-bit RNG and 48-bit keys, to support security. For reliability the transmitted messages are (only) 16-bit numbers. Each tag \mathcal{T} stores a 16-bit number P and a 48-bit key $K = K_0 || K_1$, where K_0 is 32-bits long. The server \mathcal{S} stores for each tag \mathcal{T} in a database DB an entry with two 48-bit keys K^{old}, K^{cur} , that is:

$$\langle id(\mathcal{T}), K^{old}, K^{cur} \rangle .$$

Initially P is assigned a random value, K^{cur}, K are assigned the same random value, and K^{old} is null.

TRAP-3

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N .
 \mathcal{T} computes $L = (K_1 \oplus P) || N$, draws three 32-bit numbers M_1, M_2, M_3 from $f_{K_0}(L)$, parses $M_1 = M_{10} || M_{11}$ and $M_3 = M_{30} || M_{31}$ into 16-bit numbers, and sets $M \leftarrow M_{10}$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M .
 \mathcal{T} updates $P \leftarrow M_{11}$.
 \mathcal{S} computes $L' = (K_1 \oplus P) || N$ and draws M'_1, M'_2, M'_3 from $f_{K_0}(L')$, for every $K^j, j \in \{old, cur\}$ in DB , parses $M'_1 = M'_{10} || M'_{11}$ and $M'_2 = M'_{20} || M'_{21}$, and sets $Q \leftarrow M'_{20}$.
 If for some K^j in DB there is a match $M'_{10} = M$, then \mathcal{T} is authentic: for $j = cur$ it updates $K_1^{old} \leftarrow K^{cur}, K_1^{cur} \leftarrow M'_{21} || M'_3$; for $j = old$ it updates $K^{cur} \leftarrow M'_{21} || M'_3$.
 Else \mathcal{T} is rejected.
3. $\mathcal{S} \Rightarrow \mathcal{R}$: Q , details regarding the tag, “end session”.
4. $\mathcal{R} \rightarrow \mathcal{T}$: Q .
 \mathcal{T} checks that $Q = M_{20}$; if so, it updates $K \leftarrow M_{21} || M_3$.

The security of TRAP-3. TRAP-3 extends TRAP-2 to capture forward secrecy. To prove forward secrecy we need to show that the adversary cannot (a) desynchronize the updating process of the key K of a tag \mathcal{T} and, (b) link a tag whose key is compromised to earlier protocol flows (obtained by eavesdropping).

For (a) there are two cases to consider. If the adversary is passive then the value of the key K of \mathcal{T} is the same as the value K^{cur} stored in DB for \mathcal{T} . If the adversary is active and prevents \mathcal{T} from receiving the confirmation Q , or if a rogue reader interrogates \mathcal{T} , then the value of K is K^{old} —even if such attacks are repeated. For (b) observe that if a tag’s key K is compromised then the earlier flows $[N; P, M; Q]$ cannot be linked because they are pseudo-random.

Observe that this protocol is optimistic [24]: if the server stores the values P for each tag \mathcal{T} , then these can be used to disambiguate the tag \mathcal{T} when the adversary is passive (an eavesdropper) or inactive. Finally, as with the protocols O-TRAP and O-FRAP [5,24], a compromised tag can be traced back to its last completed interrogation with an authorized reader, since it will not update its key K until it receives a (valid) confirmation Q .

5 Adding a Kill Functionality

Our TRAP protocols can be extended to include a kill feature for tags. In this section we show how this is done for TRAP-2. Observe that to disable a particular tag, the server must first authenticate that tag.

5.1 TRAP-2*: A TRAP with Kill Functionality

EPCGen2 specifies four states that tags may implement: *open*, *ready*, *secure* and *killed*. *Open* is the initial state; *ready* is a holding state until the tag receives the

next message from the reader; *secure* is a state into which a tag transitions on receiving an authenticator from the reader; *killed* is a state into which a *secured* tag transitions on receiving a kill mandate. A *killed* tag will not respond to any challenge, and cannot be resurrected.

In TRAP-2* each tag \mathcal{T} stores a 16-bit number P and a 32-bit key $K = K_0||K_1$, as in TRAP-2, but also stores an additional 32-bit *kill-key* KP . The server \mathcal{S} stores in a database DB , for each tag \mathcal{T} , an entry of the form:

$$\langle id(\mathcal{T}), K, KP, EPC, state_tag, kill_mandate \rangle,$$

where *state_tag* specifies the state of the tag and *kill_mandate* is either *null* or *kill*. Initially K and P are assigned random values, *state_tag* is *open* and *kill_mandate* is *null*.

TRAP-2*

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: A 16-bit random nonce N .
 \mathcal{T} updates its state to *ready*, computes $L = (K_1 \oplus P)||N$ and draws two 16-bit numbers M, M_1 from $f_{K_0}(L)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M .
 \mathcal{T} updates $P \leftarrow M_1$.
 \mathcal{S} computes $L' = (K_1 \oplus P)||N$ and draws two numbers M', M'_1 from $f_{K_0}(L')$ for every key K in DB .
 If there is a match $M' = M$ then \mathcal{T} is authentic: \mathcal{S} updates \mathcal{T} 's *state_tag* in DB to *secure*. Else the tag is rejected.
 If the value of *kill_mandate* of a *secure* tag \mathcal{T} is *kill*, then the server \mathcal{S} parses $KP = KP_0||KP_1$ into 16-bit numbers and draws a 16-bit number Q from $f_{KP_0}((KP_1 \oplus P)||N)$.
3. If the value of *kill_mandate* of a *secure* tag \mathcal{T} in DB is *null* then:
 $\mathcal{S} \Rightarrow \mathcal{R}$: Details regarding the tag, and “end session”.
4. If the value of *kill_mandate* of a *secure* tag \mathcal{T} in DB is *kill* then:
 $\mathcal{S} \Rightarrow \mathcal{R}$: Q , and “end session”.
 $\mathcal{R} \rightarrow \mathcal{T}$: Q .
 \mathcal{T} : If the value of *state_tag* is *ready* then \mathcal{T} parses $KP = KP_0||KP_1$ and draws a number Q' from $f_{KP_0}((KP_1 \oplus P)||N)$.
 If $Q' = Q$ then \mathcal{T} transitions to a *killed* state.

5.2 Security Analysis of TRAP-2*

We only discuss the security of the kill functionality. As noted earlier, non-corrupted tags that receive a kill mandate will always assume a *killed* state. However a tag may be prevented from receiving the kill mandate Q by the adversary. In such cases even though the value of *state_tag* in the database DB is *killed*, the tag is not *killed*. However the tag is, for all practical purposes, disabled: each time it attempts to get identified by an authorized reader, it will only receive a kill mandate.

6 Concluding Remarks

The EPCGen2 standard for Class 1 tags focuses on reliability and efficiency and supports only a very basic security level. Designing EPCGen2 compliant RFID protocols that are secure is particularly challenging because the only security tool that is available in this standard is a 16-bit RNG.

In this paper we have shown that two recently proposed EPCGen2 compliant RFID protocols fail to provide adequate security and are subject to impersonation attacks and synchronization attacks. We proposed two basic RFID authentication protocols, TRAP-1 and TRAP-2 that are EPCGen2 compliant, whose security is reduced to the minimal security levels supported by this standard, and have shown how to add a kill functionality. Finally we proposed a mutual authentication RFID protocol that provides strong anonymity and that complies with the EPC Class2 Gen2 standard.

Acknowledgement

The authors thank the anonymous reviewers for helpful comments and suggestions.

References

1. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable RFID tags via insubvertible encryption. In: Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2005), pp. 92–101. ACM Press, New York (2005)
2. Avoine, G.: <http://lasecwww.epfl.ch/~gavoine/rfid/>
3. Avoine, G., Oechslin, P.: A scalable and provably secure hash based RFID protocol. In: Proc. IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005). IEEE Computer Society Press, Los Alamitos (2005)
4. Burmester, M., de Medeiros, B., Motta, R.: Robust, Anonymous RFID Authentication with Constant Key-Lookup. In: Proc. ACM Symposium on Information, Computer and Communication Security (ASIACCS 2008), pp. 283–291. ACM Press, New York (2008)
5. Burmester, M., van Le, T., de Medeiros, B.: Provably secure ubiquitous systems: Universally composable RFID authentication protocols. In: Proceedings of the 2nd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks (SECURECOMM 2006). IEEE Press, Los Alamitos (2006)
6. Canetti, R.: Studies in Secure Multiparty Computation and Application. PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel (June 1995)
7. Canetti, R.: Security and composition of multi-party cryptographic protocols. *Journal of Cryptology* 13(1), 143–202 (2000)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proc. IEEE Symp. on Foundations of Computer Science (FOCS 2001), pp. 136–145. IEEE Press, Los Alamitos (2001)
9. Chien, H.-Y., Chen, C.-H.: Mutual authentication protocol for rfid conforming to EPC class 1 generation 2 standards. *Comput. Stand. Interfaces* 29(2), 254–259 (2007)

10. Duc, D.N., Jaemin Park, H.L., Kim, K.: Enhancing Security of EPCglobal Gen-2 RFID Tag against Traceability and Cloning (2006), Symposium on Cryptography and Information Security, SCIS 2006 (2006)
11. Dimitriou, T.: A lightweight RFID protocol to protect against traceability and cloning attacks. In: Proc. IEEE Intern. Conf. on Security and Privacy in Communication Networks (SECURECOMM 2005). IEEE Press, Los Alamitos (2005)
12. Dimitriou, T.: A secure and efficient RFID protocol that can make big brother obsolete. In: Proc. Intern. Conf. on Pervasive Computing and Communications (PerCom 2006). IEEE Press, Los Alamitos (2006)
13. EPC Global. EPC Tag Data Standards, vs. 1.3. http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf
14. Goldreich, O., Goldwasser, S., Micali, S.: How to construct pseudorandom functions. *Journal of the ACM* 33(4) (1986)
15. Henrici, D., Müller, P.M.: Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In: Proc. IEEE Intern. Conf. on Pervasive Computing and Communications, pp. 149–153 (2004)
16. ISO/IEC. Standard # 18000 – RFID Air Interface Standard, <http://www.hightechaid.com/standards/18000.htm>
17. Juels, A.: Minimalist cryptography for low-cost RFID tags. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 149–164. Springer, Heidelberg (2005)
18. Menezes, A., van Oorschot, P., Vanstone, S.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton (1996)
19. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, Springer, Heidelberg (2006)
20. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic approach to “privacy-friendly” tags. In: Proc. RFID Privacy Workshop (2003)
21. Sharma, S.E., Weiss, S.A., Engels, D.W.: RFID systems and security and privacy implications. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 454–469. Springer, Heidelberg (2003)
22. Stinson, D.: *Cryptography Theory and Practice*, 2nd edn. CRC Press, Inc., Boca Raton (2002)
23. Tsudik, G.: YA-TRAP: Yet another trivial RFID authentication protocol. In: Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2006), IEEE Press, Los Alamitos (2006)
24. van Le, T., Burmester, M., de Medeiros, B.: Universally composable and forward-secure RFID authentication and authenticated key exchange. In: Proc. of the ACM Symp. on Information, Computer, and Communications Security (ASIACCS 2007), pp. 242–252. ACM Press, New York (2007)
25. Weis, S., Sarma, S., Rivest, R., Engels, D.: Security and privacy aspects of low-cost radio frequency identification systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) *Security in Pervasive Computing*. LNCS, vol. 2802, pp. 454–469. Springer, Heidelberg (2004)

Appendix: The Security of TRAP-2

We first describe the protocol O-TRAP [5] and show the modifications needed to get TRAP-2. We then state our main result and briefly discuss the security framework.

In O-TRAP, each tag \mathcal{T} stores two n -bit numbers: a pseudonym P and a key K , where n is the security parameter. The server \mathcal{S} has two databases, DB and DB' : for each tag \mathcal{T} , the server \mathcal{S} stores in DB an entry of the form $\langle id(\mathcal{T}), K \rangle$, and in DB' an entry of the form $\langle id(\mathcal{T}), P, K \rangle$. DB is indexed by the (authorized) keys K while DB' is indexed by the pseudonyms P . Initially P and K are assigned random values. Let $H_K(\cdot)$ be a hash function with pseudorandom values.

O-TRAP

1. $\mathcal{S} \Rightarrow \mathcal{R} \rightarrow \mathcal{T}$: An n -bit random number N .
 \mathcal{T} computes $M = H_K(N||P)$.
2. $\mathcal{T} \rightarrow \mathcal{R} \Rightarrow \mathcal{S}$: P, M .
 \mathcal{T} updates $P \leftarrow M$.
 \mathcal{S} accepts the tag \mathcal{T} as authentic if:
 either there exists an entry $(id(\mathcal{T}), P, K) \in DB'$ with $M = H_K(R_{sys}||P)$,
 or there exists an entry $(id(\mathcal{T}), K) \in DB$ with $M = H_K(R_{sys}||P)$.
 Else the tag is rejected.

We have:

Theorem 1. [5] *O-TRAP guarantees availability, anonymity, and authentication in the Universal Composability (UC) framework [6,7,8] provided the keyed hash function is chosen from a pseudo-random function family $\{H_K(\cdot)\}_{|K|=n}$.*

A key feature of this protocol is that it is *optimistic*, that is its security overhead is minimal when the adversary is passive (an eavesdropper) or inactive, since in this case the server \mathcal{S} needs to do only one key-lookup in DB' to find the pseudonym of \mathcal{T} and then authenticate the tag (we have constant key-lookup [4]). TRAP-2 is not optimistic, however in all other respects is very similar to O-TRAP. For both protocols the pseudonyms P and authenticators M are pseudo-random. In TRAP-2, M is drawn from $f_{K_0}(K_1 \oplus P||N)$, whereas in O-TRAP it is $H_K(P||N)$. Consequently M is a pseudo-random number determined by: the key K , the pseudonym P , and the challenge P . It follows that one can use the same steps as in the security proof for O-TRAP to get a proof for TRAP-2. We need however to make certain that $f_{K_0}(K_1 \oplus \cdot || \cdot)$ is a PRF, which in our case is guaranteed if the length of the seed is sufficiently long to prevent the adversary from distinguishing its output from random numbers.

The UC-framework. UC security is based on notions of interactive indistinguishability of real from ideal protocol executions. This requires:

1. A mathematical model of real protocol executions, where honest parties are represented by probabilistic polynomial-time Turing machines that correctly execute the protocol as specified, and adversarial parties that can deviate from the protocol in an arbitrary way. The adversarial parties are controlled by a single (PPT) adversary \mathcal{A} that (1) has full knowledge of the state of adversarial parties, (2) can arbitrarily schedule the actions of all parties, both honest and adversarial, and (3) interacts with the environment in arbitrary ways, in particular can eavesdrop on all communications.

2. An idealized model of protocol executions, where the security properties are defined in terms of an *ideal functionality* \mathcal{F} , a trusted party that all parties may invoke to guarantee correct execution of particular protocol steps. The ideal-world adversary $\mathcal{S}_{\mathcal{A}}$ is controlled by the ideal functionality, to reproduce as faithfully as possible the behavior of the real adversary.
3. A proof that no environment can distinguish (with better than negligible probability) real- from ideal-world protocol runs by observing the system behavior.

In the real world the adversary \mathcal{A} interacts with the protocol parties using commands such as REFRESH, START, SEND and END, to cause the beginning of a new server interrogation period, to make available a tag for interrogation, to send a challenge to a tag and get its response, or to send a response to the server, etc. The goal of \mathcal{A} is to prevent an honest tag from (i) getting accepted by the server (availability), (ii) getting authenticated, and (iii) being anonymous. The ideal adversary $\mathcal{S}_{\mathcal{A}}$ has the same goals, but this time the interactions are with the ideal-world functionality \mathcal{F} . We get UC-security if no PPT environment \mathcal{Z} can distinguish real- from ideal-world simulations. For more details the reader is referred to [5,24].

Author Index

- Au, Man Ho 94
- Baek, Joonsang 75
Bao, Feng 75
Batina, Lejla 446
Blanton, Marina 188
Borisov, Nikita 373
Burmester, Mike 490
- Canard, Sébastien 207, 258
Canright, D. 446
Chen, Chia-Hsin Owen 242
Chen, Ming-Shing 242
Cheng, Chen-Mou 242
Cvrcek, Dan 460
- Damgård, Ivan B. 144
de Medeiros, Breno 490
Ding, Jintai 242
Dodis, Yevgeniy 156
- Ferrer-Gomila, Josep Lluís 174
Fouque, Pierre-Alain 411
- Gouget, Aline 207
- Hammouri, Ghaith 346
Huang, Qiong 94
Huguet-Rotger, Llorenç 174
- Keromytis, Angelos D. 39
Kim, Kwangjo 224
King, Brian 429
Klonowski, Marek 296
Knudsen, Lars R. 144
Kutyłowski, Mirosław 296
- Laguillaumie, Fabien 258
Lakshminarayanan, A. 392
Lauks, Anna 296
Lewis, Matt 460
Li, Jin 224
Lim, Tong-Lee 392
Liu, Joseph K. 94
Locasto, Michael E. 39
- Martinet, Gwenaëlle 411
Milhau, Michel 258
Mitchell, John C. 309
Mitra, Soumyadeb 373
Monrose, Fabian 21
Mut-Puigserver, Macià 174
- Nishide, Takashi 111
- Ohta, Kazuo 111
Ouafi, Khaled 479
- Payeras-Capellà, Magdalena 174
Phan, Raphael C.-W. 479
Pointcheval, David 277
Prasad, Ramnath 328
Provos, Niels 21
Puniya, Prashant 156
- Rajab, Moheeb Abu 21
Rowe, Paul 309
Roy, Arnab 309
- Saksen, Vira 392
Sanadhya, Somitra Kumar 130
Sarkar, Palash 130
Saxena, Nitesh 328
Scedrov, Andre 309
Shpilrain, Vladimir 366
Smith, Sean W. 55
Song, Hui 1
Stajano, Frank 460
Stavrou, Angelos 39
Sunar, Berk 346
Susilo, Willy 94
- Terzis, Andreas 21
Thomsen, Søren S. 144
Tsang, Patrick P. 55
- Ushakov, Alexander 366
- Valette, Frédéric 411
- Wong, Duncan S. 94, 224

Xie, Liang 1

Yang, Bo-Yin 242

Yang, Guomin 94

Yoneyama, Kazuki 111

Zhang, Fanguo 224

Zhou, Jianying 75

Zhu, Suncun 1

Zimmer, Sébastien 277, 411