Liqun Chen
Mark D. Ryan
Guilin Wang (Eds.)

LNCS 5308

Information and Communications Security

10th International Conference, ICICS 2008
Birmingham, UK, October 2008
Proceedings

Springer

Lecture Notes in Computer Science 5308

Liqun Chen   Mark D. Ryan   Guilin Wang (Eds.)

Information and Communications Security

10th International Conference, ICICS 2008
Birmingham, UK, October 20 - 22, 2008
Proceedings

Springer

Volume Editors

Liqun Chen
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK
E-mail: liqun.chen@hp.com

Mark D. Ryan
Guilin Wang
University of Birmingham
School of Computer Science
Edgbaston, Birmingham B15 2TT, UK
E-mail: {m.d.ryan, g.wang}@cs.bham.ac.uk

Library of Congress Control Number: 2008936716

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-88624-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-88624-2 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12539420 06/3180 5 4 3 2 1 0

Preface

The 10th International Conference on Information and Communications Security (ICICS) was held in Birmingham, UK, during 20–22 October 2008. The ICICS conference series is an established forum that brings together people working in different fields of information and communications security from universities, research institutes, industry and government institutions, and gives the attendees the opportunity to exchange new ideas and investigate state-of-the-art developments. In previous years, ICICS has taken place in China (2007, 2005, 2003, 2001, 1997), USA (2006), Spain (2004), Singapore (2002), and Australia (1999). On each occasion, as on this one, the proceedings were published in the Springer LNCS series.

In total, 125 papers from 33 countries were submitted to ICICS 2008, and 27 were accepted covering multiple disciplines of information security and applied cryptography (acceptance rate 22%). Each submission to ICICS 2008 was anonymously reviewed by three or four reviewers. We are grateful to the Programme Committee, which was composed of 57 members from 12 countries; we thank them as well as all external referees for their time and valued contributions to the tough and time-consuming reviewing process.

In addition to the contributed speakers, the programme also featured three invited speakers. We are grateful to Joshua Guttman (The MITRE Corporation, USA), Peng Ning (North Carolina State University, USA), and Nigel Smart (University of Bristol, UK) for accepting our invitation to speak.

ICICS 2008 was organised by the University of Birmingham and Hewlett Packard Laboratories. We gratefully acknowledge sponsorship from the UK Engineering and Physical Sciences Research Council (EPSRC), as well as Hewlett Packard and the University of Birmingham.

Organising a conference is difficult and time-consuming work. We are very grateful to Andy Brown, who worked tirelessly in making arrangements with the Hyatt Hotel and with the University of Birmingham, as well as maintaining the conference website. Also thanks to Ben Smyth for making a great job of collecting the papers together for these proceedings. Thanks also to Hasan Qunoo for helping with many local details. Finally, we would like to thank all the authors who submitted their papers to ICICS 2008, and all the attendees from all over the world.

October 2008

Liqun Chen
Mark Ryan
Guilin Wang

Organisation

ICICS 2008

10th International Conference on Information and Communications Security

Birmingham, UK
October 20–22, 2008

Organised by

School of Computer Science, University of Birmingham, UK

Sponsored by

Engineering and Physical Sciences Research Council (EPSRC)
Hewlett-Packard Laboratories
University of Birmingham

In co-operation with

International Communications and Information Security Association (ICISA)

General Chair

Mark Ryan	University of Birmingham, UK

Programme Chairs

Liqun Chen	Hewlett-Packard Laboratories, UK
Mark Ryan	University of Birmingham, UK
Guilin Wang	University of Birmingham, UK

Programme Committee

Mikhail Atallah	Purdue University, USA
Tuomas Aura	Microsoft Research, UK
Vijay Atluri	Rutgers University, USA
Michael Backes	Saarland University, Germany
Feng Bao	Institute for Infocomm Research, Singapore
Elisa Bertino	Purdue University, USA
Alex Biryukov	University of Luxembourg, Luxembourg
Colin Boyd	Queensland University of Technology, Australia
Srdjan Capkun	ETH Zurich, Switzerland

Chin-Chen Chang	Feng Chia University, Taiwan
Hao Chen	University of California at Davis, USA
Kefei Chen	Shanghai Jiaotong University, China
Edward Dawson	Queensland University of Technology, Australia
Robert Deng	Singapore Management University, Singapore
Dengguo Feng	Chinese Academy of Science, China
Steve Furnell	University of Plymouth, UK
Dieter Gollmann	Hamburg University of Technology, Germany
David Grawrock	Intel, USA
Hongxia Jin	IBM Almaden Research Center, USA
Engin Kirda	Institute Eurecom, France
Steve Kremer	ENS de Cachan, France
Chi-Sung Laih	National Cheng Kung University, Taiwan
Dong Hoon Lee	Korea University, Korea
Ninghui Li	Purdue University, USA
Qun Li	College of William and Mary, USA
Yingjiu Li	Singapore Management University, Singapore
Javier Lopez	University of Malaga, Spain
Wenbo Mao	EMC Research, China
Catherine Meadows	Naval Research Laboratory, USA
Chris Mitchell	Royal Holloway, UK
Sang-Jae Moon	Kyungpook National University, Korea
Yi Mu	University of Wollongong, Australia
Peng Ning	North Carolina State University, USA
Eiji Okamoto	University of Tsukuba, Japan
Akira Otsuka	AIST, Japan
Kenneth Paterson	Royal Holloway, UK
Giuseppe Persiano	Università di Salerno, Italy
Raphael Phan	Loughborough University, UK
Sihan Qing	Chinese Academy of Sciences, China
Kui Ren	Illinois Institute of Technology, USA
Eike Ritter	University of Birmingham, UK
Bimal Roy	Indian Statistical Institute, India
Peter Ryan	University of Newcastle, UK
Kouichi Sakurai	Kyushu University, Japan
Steve Schneider	University of Surrey, UK
Wenchang Shi	Renmin University of China, China
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University Hakodate, Japan
Bogdan Warinschi	Universtiy of Bristol, UK
Andreas Wespi	IBM Zurich Research Laboratory, Switzerland
Duncan S. Wong	City University of Hong Kong, China
Yongdong Wu	Institute for Infocomm Research, Singapore
Alec Yasinsac	Florida State University, USA
Moti Yung	Columbia University, USA

Yuliang Zheng	University of North Carolina at Charlotte, USA
Jianying Zhou	Institute for Infocomm Research, Singapore
Sencun Zhu	Penn State University, USA

Organising Committee

Andrew Brown	University of Birmingham, UK
Hasan Qunoo	University of Birmingham, UK
Ben Smyth	University of Birmingham, UK
Guilin Wang	University of Birmingham, UK

Publication Chairs

Ben Smyth	University of Birmingham, UK
Guilin Wang	University of Birmingham, UK

External Reviewers

Isaac Agudo
Cristina Alcaraz
Man Ho Au
Jean-Philippe Aumasson
Vicente Benjumea
Shaoying Cai
Giacomo Cancelli
Jianhong Chen
Carlos Cid
Andrew Clark
Nathan Clarke
Yvonne Cliff
Xuhua Ding
Rong Du
Markus Duermuth
Marie Duflot
Serge Fehr
Marcel Fernandez
Carmen Fernandez-Gago
Ernest Foo
Clemente Galdi
Juan Gonzalez
Jae-Cheol Ha
Manabu Hagiwara
Keisuke Hakuta
Hao Han
Wei Han
Keith Harrison
Nick Hoare
Xuan Hong
Yoshiaki Hori
Cătălin Hriţcu
Qiong Huang
Xinyi Huang
Jung Yeon Hwang
Manabu Inuma
Yoon-Chan Jhi
Qingguang Ji
Ashish Kamra
Takashi Kitagawa
Shinsaku Kiyomoto
Ilya Kizhvatov
Boris Köpf
Ji-Seon Lee
Fagen Li
Gaicheng Li
Jun Li
Bing Liang
Huo-Chong Ling
Joseph Liu
Yu Long
JiQiang Lu
Di Ma
Wenbo Mao
Miodrag Mihaljevic
George Mohay
Ian Molloy
Shiho Moriai
Qun Ni
Chihiro Ohyama
Elisabeth Oswald
Maria Papadaki
Jong Hwan Park
Serdar Pehlivanoglu
Jason Reid
Mohammad-Reza Reyhanitabar
Bo Sheng
Nicholas Sheppard
SeongHan Shin
Taizo Shirai
Leonie Simpson
Hui Song
Graham Steel
Makoto Sugita
Ashwin Swaminathan
Chiu Tan
Hitoshi Tanuma

Alberto Trombetta
Ivan Visconti
Yongtao Wang
Xinran Wang
Haodong Wang
Qihua Wang
Gaven Watson
Jian Weng
Zhe Xia
Liang Xie
Fengyuan Xu
Guomin Yang
Yanjiang Yang
Ng Ching Yu
Rui Zhang
Jinmin Zhong
Jakub Zimmermann

Table of Contents

Invited Talk

Authentication

Side Channel Analysis

Cryptanalysis

Access Control

Software Security

System Security

Applied Cryptography

Security Protocols

Attestation: Evidence and Trust

George Coker, Joshua Guttman, Peter Loscocco, Justin Sheehy,
and Brian Sniffen

The MITRE Corporation
{guttman,justin,bsniffen}@mitre.org
National Security Agency
{gscoker,loscocco}@nsa.gov

Abstract. Attestation is the activity of making a claim about properties of a target by supplying evidence to an appraiser. We identify five central principles to guide development of attestation systems. We argue that (i) attestation must be able to deliver temporally fresh evidence; (ii) comprehensive information about the target should be accessible; (iii) the target, or its owner, should be able to constrain disclosure of information about the target; (iv) attestation claims should have explicit semantics to allow decisions to be derived from several claims; and (v) the underlying attestation mechanism must be trustworthy. We propose an architecture for attestation guided by these principles, as well as an implementation that adheres to this architecture. Virtualized platforms, which are increasingly well supported on stock hardware, provide a natural basis for our attestation architecture.

1 Introduction

Much economic activity takes place on heterogeneous networks of computers, involving interactions among autonomous principals, including individuals, retail companies, credit card firms, banks, and stock brokerages. Because the amount of money in these activities is large and increasing, the networks are attractive targets for criminals.

In many attacks, the adversary inserts software remotely, without physical access to the devices, and this software compromises secrets. For instance, in March 2008, an attack was announced against the large American grocery store chain Hannaford Brothers. Unauthorized code had been inserted on the servers in each of the company's 300 stores. This code retained the credit card information for each transaction occurring at a store, and periodically transmitted the information to a third party. As a consequence, over 4,200,000 credit and debit cards were compromised. At least 2,000 fraudulent transactions have been identified as results. Even though Hannaford's systems were designed not to store customer payment details, and to adhere to compliance standards of the credit card companies, changes to their application software led to large disclosures [15].

An even larger case led to indictments in August 2008. Over 40 million card numbers were stolen from US companies such as TJX, a clothing distributor

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 1–18, 2008.

and retailer, and other large firms. According to the indictment papers, eleven criminals collaborated in this group of attacks. Members were located in the US, Estonia, Ukraine, Belarus, and China. In these attacks, wireless access points were the initial entry point. Newspapers described the inserted software as sniffers. However, the indictments mention that an insecure wireless access point at a Marshall's retail store in Florida allowed the defendants to compromise data stored in servers at TJX, located in Massachusetts [24].

In both of these cases, the inserted software appears to have remained undetected for months.

There are three characteristics of these attacks. First, the attacks are executed remotely, apparently without physical access to the computers attacked. Second, the computers are standard, general purpose systems, rather than specialized devices such as automated teller machines. Third, the networks involve transactions among independent organizations, such as a retailer, a distributor, and the credit card firms. No one organization controls the software configurations on all the relevant systems. The ubiquitous attacks that insert malware onto individually owned computers, to sniff for bank account and password information, share these characteristics. The bank cannot control the configurations of its customers' computers. Nevertheless, there would be benefits shared by the bank and its customers if the bank could ascertain that the customer's computer was free of malware, before allowing the customer to enter the account number and password.

Attestation means providing reliable evidence about the state of software executing on a system. Many computing problems could be solved if attestation were reduced to practice, particularly attestation that provides evidence of the behavioral properties similar to those mentioned in the attacks we have mentioned. To achieve this goal, attestation must make sense for general-purpose systems, running varied configurations, and under the control of different individuals and organizations. The participant's privacy goals must be respected, while providing evidence that distributed transactions are not being subverted.

One might think that attestation in this sense would be impossible. What evidence of a good state can be provided to a remote party, that could not be synthesized by bad software? We do not—by analogy—ask someone we suspect of being a scam artist whether he is honest, or at least we do not count it as evidence when he says that he is. However, a starting point for trust appraisal now exists, in the form of the Trusted Platform Module (TPM), introduced by the Trusted Computing Group (TCG) [2]. The TPM provides primitive cryptographic capabilities, and some long-term storage that is secure from remote attacks, which can be used to provide signed evidence from Platform Configuration Registers that reflect key events of the software controlling the machine.

Existing attestation proposals, including those put forth by the TCG, are generally aimed at specific use-cases and typically lack flexibility to address a more general attestation problem. Further, existing *definitions* of attestation primarily focus on describing the particular properties [19] desirable in those use-cases. For example, in [7], the author uses the term attestation to specifically mean the process of transmitting a sequence of hashes of certain system components and a digital signature of that sequence; in Microsoft's "NGSCB" [6] it refers to identification and authentication of known code via digital signatures; Copilot [13] makes use of direct hashes of kernel memory, and so on. We prefer a general definition of platform attestation that abstracts from specific desired properties [3].

In this paper, we describe a flexible attestation architecture, based on a few guiding principles. Systems built according to this architecture can be composed to carry out attestation scenarios. We believe that this attestation architecture provides the mechanisms needed for systems to interrogate each other before sensitive interactions, so as to ensure that those interactions will be safe.

2 Attestation

Our approach to system attestation departs significantly from the notion put forth by the TCG, in great part due to increased flexibility. Emphasis is placed on attestation based upon *properties* of the target, useful in a variety of scenarios, rather than solely on attestation based upon *identity.*

Terminology. *An* appraiser *is a party, generally a computer on a network, making a decision about some other party or parties. A* target *is a party about which an appraiser makes such a decision.*

The trust decision made by an appraiser often supports an access request made on behalf of the target, and is usually a decision about the expected behavior of that target. To make a decision on this basis, a diligent appraiser needs a significant amount of information—essentially, the knowledge that the state of the target is such that it will not transition into an unacceptable state while the appraiser still continues to trust it. There is some inevitable tension between the human organizations behind the appraiser and target, as the appraiser's owner wishes to have complete and correct information about any given target while the target's owner wishes to give up no more than the minimal information necesssary for the success of its request (and perhaps even less).

Terminology. Attestation *is the activity of making a claim to an appraiser about the properties of a target by supplying evidence which supports that claim. An* attester *is a party performing this activity. An appraiser's decision-making process based on attested information is* appraisal.

In the most commonly addressed class of attestations, each attestation provides a means for appraisers to infer that the target of the attestation will not engage in a class of misbehaviors. For example, if the target reports its kernel is unmodified, the attester has reason to trust reports from the target, and some appraiser trusts information provided by the attester, then that appraiser can infer that the target will not engage in misbehaviors that might have occurred had the target's kernel been corrupted at the time of its measurement. It is important to note that not all attestations are about lack of misbehaviors, even though most of the commonly discussed use cases are in that class.

This broader point of view makes a rich understanding of the related concepts of system measurement, attestation protocols, and system separation vital to successful attestation. Here there is a distinction between the measurement of a target system (the evidence) and the attestation itself.

Terminology. *To* measure *a target means to collect evidence about it through direct, local observation.*

Attestation about a target system will report measurements or conclusions inferred using measurements and possibly also other attestations. In this paper, measurement is discussed only as necessary to support our architecture for attestation.

Terminology. *An* attestation scenario *is a cryptographic protocol involving a target, an appraiser, and possibly other principals serving as trust proxies. The Trusted Platform Module residing on the target may also be considered a principal in an attestation scenario. The purpose of an attestation scenario is to supply evidence that will be considered authoritative by the appraiser, while respecting privacy goals of the target (or its owner).*

Evidence may be attested to in a number of equivalent but semantically different forms depending on the attestation scenario. For example, the attestation may report raw evidence as directly observed, as reduced evidence (e.g. a hash of the raw evidence), or by substitution with a credential provided by a third party evaluator of the raw evidence. For example, an SSL certificate authority consumes many attestations as to the identity and practices of a target, then produces a certificate attesting to the quality of a target [3].

Also, a given target may wish to provide different information to different appraisers depending on the current trust relationships it has with those parties. A worthwhile desire in developing an attestation system is to resolve the mutual tension as well as posssible given the contradictory nature of the parties' interests. One approach to defusing this tension is for the appraiser to demand frequent repeated attestations, re-evaluating its trust decisions often. It may be possible to determine that a party will be sufficiently trustworthy for the 15 minutes after performing a given attestation, but not feasible to determine that it will be so for a day.

3 Principles for Attestation Architectures

Five principles are crucial for attestation architectures. While an ideal attestation architecture would satisfy all five, they may not all be satisfiable in some kinds of systems. Thus, attestation mechanisms may emphasize some features over others. The five principles motivate the architecture presented in Section 4.

Principle 1 (Fresh information). *Assertions about the target should reflect the* running system*, rather than just disk images. While some measurement tools may provide start-up time information about the target, others will inspect the* *current state of an active target. An attestation architecture should ensure access to the live state of the target.* □

The architecture cannot predict the uses to which appraisers will put the information it delivers. Appraisers may need to make very different decisions, and—to justify them—need to make different predictions about the future behavior of the target. This suggests the next principle.

Principle 2 (Comprehensive information). *Attestation mechanisms should be capable of delivering comprehensive information about the target, and its full internal state should be accessible to local measurement tools.* □

With comprehensive information come worries about the consequences of disclosure. Disclosure may cause loss of privacy for a person using the target platform. It can also subject the platform to attack, for instance if the attestation discloses an unpatched vulnerability to an adversary.

Principle 3 (Constrained disclosure). *Targets should be able to enforce policies governing which measurements are sent to each appraiser.* □

Hence, an attestation architecture must allow the appraiser to be identified to the target. Policies may distinguish kinds of information to be delivered to different appraisers. The policy may be dynamic, relying on current run-time information for individual disclosure decisions. For instance, a target may require that the appraiser provide an attestation of its own state, before the target discloses its own.

Principle 4 (Semantic explicitness). *The semantic content of attestations should be explicit.* □

The identity of the target should be included, so an appraiser can collect attestations about it. The appraiser should be able to infer consequences from several attestations, e.g. when different measurements of the target jointly imply a prediction about its behavior. Hence, attestations should have uniform semantics, and so that they are composable using valid logical inferences.

Principle 5 (Trustworthy mechanism). *Attestation mechanisms should provide evidence of their trustworthiness. In particular, the attestation architecture in use should be identified to both appraiser and target.* □

There will be a good deal of natural variation in how different systems meet these principles, and in the choices they make when some principles are only partially satisfied. In specific situations, it may be sufficient to satisfy these principles only partly. For instance, limited forms of evidence about the target may suffice for an appraiser, or evidence that has aged to an extent may be accepted. When different degrees of adherence to the principles are designed into a system, then the variation is static. When the system adjusts at runtime to provide different degrees of evidence in different situations, or when different peers are the appraiser, then the variation is dynamic.

4 Proposed Attestation Architecture

There are five main constraints, imposed by the principles of Section 3, that provide the content for the proposed architecture. In this section, each constraint is briefly described in the context of how it is motivated by the principles. A system designed according to this architecture must have the following abilities:

1. To *measure* diverse aspects of the target of attestation;
2. To *separate domains* to ensure that the measurement tools can prepare their results without interference from the (possibly unreliable) target of attestation;
3. To *protect itself*, or at least a core *trust base* that can set up the domain separation mechanism, ensuring that it cannot be weakened without this fact being evident from the content of attestations;
4. To *delegate attestation* so that attestation proxies can collect detailed measurements and convincing evidence, and summarize them to selected peers, when the target would not permit the full facts to be widely shared;
5. To *manage attestation* to handle attestation queries by invoking suitable measurement tools, delivering the results to the appraiser or a proxy as constrained by policies.

These constraints are discussed in turn.

4.1 Measurement Tools

Providing comprehensive information about a system (satisfying Principle 2) requires the ability to provide a collection of tools that (jointly) comprehensively measure the target.

Comprehensive measurement of a system requires more than simply the ability to read all of the data contained in that system. It also means that some measurement tools must understand the structure of what they are measuring. For example, just being able to scan and hash the memory used by an operating system kernel may not suffice to provide useful measurements of it. *Usefulness*, here, is in the eye of the appraiser, and typically involves evidence about the past or future *behavior* of the target. The state of a program changes during execution, and therefore cannot be measured by simple hashing. For this reason, measuring complex system components requires knowing the structure of the targets. Some trust decisions require these structure-sensitive measurements.

As a result of this, there cannot be a “one size fits all” measurement capability for attestation. Different measurement tools must be produced for measuring components with different structure. Further, the complete set of such tools that will be desired is not knowable ahead of time without restricting the target systems from ever adding any new future applications.

Our architecture must support a collection of specialized measurement tools, and in order to be able to provide evidence for arbitrary future attestations it must also support adding new tools to that collection over time.

In addition to measurement capacity being comprehensive, freshness is also a goal. (Principle 1) This means that our measurements cannot always be performed a priori – one must be able to measure various parts of a system on demand. These demands are made from the point of view of an appraiser. A remote party must be able to trigger measurement; it is insufficient to only have runtime measurement occur via periodic automatic remeasurement triggered by the measurement system or tools.

4.2 Domain Separation

For a measurement tool to provide information about a target of attestation, the measurement tool must be able to deliver accurate results even when the target is corrupted. This is an important consequence of Principle 5.

There are two parts to this. First, it must have access to the target's state so as to be able to distinguish whether that target is corrupted or uncorrupted. This state includes the target's executable code but also modifiable data structures that determine whether its future behavior will be acceptable. Second, the measurement tool's state must be inaccessible to the target, so that even if the target is corrupted, it cannot interfere with the results of the measurement.

There are different ways that this separation can be achieved. One is to virtualize the target, so that the measurement tool runs in a separate virtual machine (VM) from the target [12]. The virtual machine monitor must then be able to control cross-VM visibility so that the measurement tool has read access to the target. It must also ensure that the target does not have any control over the measurement tool. There may be a message-passing channel established between them, but the hypervisor/VMM must be able to ensure transparent visibility of the measurement tool into the target and protection of those tools from the target.

Alternatives are possible. For instance, CoPilot (Section 7) uses a form of hardware separation in which the measurement tool runs on a coprocessor and the visibility constraints are expressed via hardware instead of being based on the configuration of a hypervisor.

Given the improved virtualization facilities that new processors from Intel and AMD provide, the VM approach seems like a natural approach that makes minimal requirements beyond standard commodity hardware.

4.3 Self-protecting Trust Base

We have established that domain separation is necessary in order to have trust in attestations and specifically in the integrity of our measurement tools. This raises a question: how to produce assurance for the integrity of the domain separation itself?

The core of our system's trust must come from components which are simple enough or sufficiently evaluated that one can be convinced that they do not require remeasurement after they have been running. Part of this core must obviously include the hardware used as our Trusted Computing Base. Any other component must either be measurable from a place that it cannot control or must be sufficiently measured via a static startup measurement taken before it begins operating.

Note that what is needed here is more than just a trusted static subset of our system. The difficulty here is that our trust base must be sufficiently simple and static to not require remeasurement, but also sufficiently rich to bootstrap our measurements and attestations. Anything performing measurement and attestation on the platform will in turn require measurement and attestation about itself in order to convince an appraiser of its trustworthiness. It must be ensured that this chain "bottoms out" at something sufficient to perform certain essential measurements and attestations to support the chain above it while being simple enough that static startup-time measurements are sufficient to determine trust.

It is not trivial to determine the content of this static trust base. One of the difficulties arises around the element of domain separation. It is preferable for the domain separation mechanism to be simple and secure enough to belong in this element, but no hypervisor exists today that satisfies those properties and is also rich enough to provide the services desired. This difficulty is addressed in Section 6. One possible alternative is that a hardware component provides runtime measurements of the domain separation mechanism.

4.4 Attestation Delegation

In practice, the appraiser will need to delegate many aspects of determining the quality of the target to specialists called *attestation proxies*. There are two essential reasons for this.

First, Principle 2 contains an intrinsic conflict with Principle 3. The former states that comprehensive insight into the state of the target must be available. The latter says that the target should be able to choose and enforce a policy on the disclosure of information about its state.

A natural way to reconcile these two principles is to allow appraiser and target to agree on an attestation proxy that is partially trusted by each [3]. The target trusts the proxy to disclose only information about its state which is of limited sensitivity. The appraiser trusts the proxy to make statements only when they are warranted by appropriately fresh and comprehensive information about the target.

The second reason why attestation proxies are important is that they can function as *specialists*. Enormous expertise is needed to interpret detailed measurements, such as those needed to predict behavioral properties about an operating system. An appraiser may get more reliable information and more usable information from an attestation proxy than it would be able to extract on its own from the comprehensive data. The maintainers of an attestation proxy can ensure that it has up-to-date information about the strengths and weaknesses of specific system versions or configurations.

Naturally, these delegations require protocols that allow the principals to ensure they are communicating with appropriate proxies. These protocols must supply the principals with messages that unambiguously answer the principals' questions. The design of such *attestation protocols* may follow the methods of the strand space theory [9], and may use the strand space/trust management connection from [11,10].

These delegations, combined with attestations satisfying Principle 4, enable a powerful new capability. An appraiser may compose separate layered or orthogonal attestations, involving different proxies, in order to make a judgment. Another approach, "Layering Negotiations [14]," has been proposed for flexible and composable attestation. We have used many of the same tools as this work, such as Xen and SELinux. The layered negotiations have a fixed two-level structure and are intended to enable distributed coalitions. Our approach is intended to enable general, arbitrarily flexible composability regardless of application usage model.

4.5 Attestation Management

A goal of our architecture is flexibility. It is essential that our system be able to respond meaningfully to different requests from different appraisers without having pre-arranged what every possible combination of attestations might be.

One way to support this notion is with an architectural element referred to as the *Attestation Manager*. A component realizing this idea contains a registry of all of the measurement and attestation tools currently on the platform, and a description of the semantic content produced by each. As a consequence of Principle 4, this component can select at runtime the appropriate services needed to answer any query which could be answered by some subset of the measurement and attestation capabilities currently on the system.

As an Attestation Manager will clearly be involved in nearly every remote attestation, it is also a natural place to enforce some of the constrained disclosure called for by Principle 3. It can restrict the services it selects based on the identity of the party the information would be released to, according to locally-stored access policies. In order to defend this capability from both the untrusted target of attestations and also from potentially-vulnerable measurement tools, the Attestation Manager must be protected via domain separation.

Our attestations will have to use cryptography in order to protect communications from adversaries. This same protection, taken together with domain separation, means that the target can be safely used as an intermediary for communication with appraisers or proxies. This leads to the very beneficial result that an Attestation Manager can be a purely local service; it does not need to be directly accessible by any remote parties.

4.6 The Elements of the Architecture

One might envision the elements of our architecture fitting together conceptually like so:

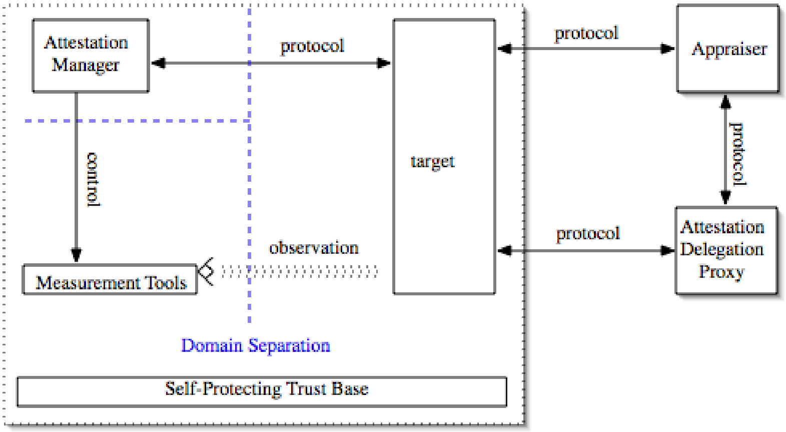


Fig. 1. Attestation Architecture

5 Composable Attestation Platform

An implementation of our attestation architecture has been developed and is illustrated in Figure 2. The hypervisor, together with the CPU, serves as the self-protecting trust base; however, the representation here is abstract, as the implementation is not tied to features specific to any one virtual machine monitor or microprocessor. The Supervisor guest (S guest) contains the platform support package, while the User guest (U guest) runs the user's "normal" operating system. The TPM hardware resource has been virtualized ("vTPM") to provide TPM capabilities for both the S and U environments. Both environments possess measurement and attestation services ("M&A").

The construction and operation of the hypervisor and each guest coincides with the collection of evidence reportable in attestations of the platform. The reason for multiple separate M&A capabilities is that evidence reported from a single party may not be sufficient.

To manage a diversity of measurement and attestation requirements, a virtual machine is dedicated to measurement and attestation (M&A) of a guest. The hypervisor is integral to the trust base for the system, controlling sharing and providing domain separation. Additional domain separation and controlled sharing requirements are met by instrumenting the M&A on SELinux [16]. The separation of the M&A and guest environments is transparent to the target and the attestation. Upon receiving attestation requests, the guest directs the incoming request to its M&A and proxy responses from that M&A back out to appraisers. To deeply assess the platform, one may need to connect attestations together across the S and U environments. This need can only be satisfied with semantically explicit attestations as described by Principle 4.

An M&A consists of three components: attestation manager (AM), attestation protocols (APs), and attestation service providers (ASPs) The AM manages the attestation session, listening for incoming attestation requests and using a "selector" subcomponent for initiating APs. An AP is a running instance of an attestation protocol initiated by the Selector in response to an attestation request. The ASPs are subcomponents of the attestation protocol. Each ASP performs a well-defined service in the attestation protocol and as defined serve a critical role in satisfying Principles 1- 5 for the platform. Some example ASPs are integrity measurement systems, wrappers for calls to a TPM/vTPM, or invocation of other services. As a result of separating these key services into ASPs which may be used by different APs, and abstracting over APs using the AM, we gain an extensible system with the ability to add new services and protocols without the need to redesign or re-evaluate the entire system for each addition.

The Selector is the mechanism for enforcing the policy of the client by instantiating APs and ASPs that conform to the policy for a given scenario thereby satisfying Principle 3. The implementation uses a method referred to as "Call by Contract" [17] for the Selector.

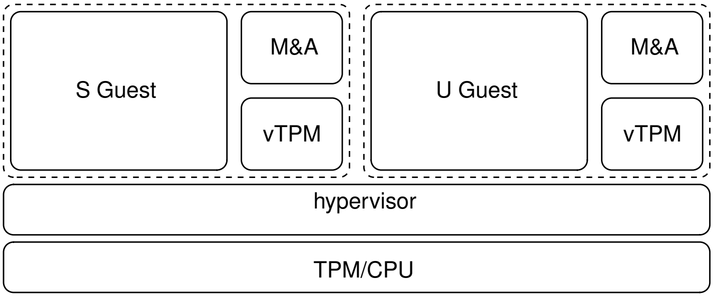


Fig. 2. Composable Attestation Platform

Attestations may be chained across the platform by the use of ASPs that make attestation requests to the other M&A environments and relay or use the attestation responses. Figure 3 shows a possible set of components that might be used in an attestation, including an ASP in the User M&A which makes an attestation request to the Supervisor M&A, enabling attestations which satisfy Principle 5.

The attestation research to date has focused exclusively on the attestation of the User OS kernel and the core platform (the Supervisor guest and hypervisor components). The attestation of these components forms the trust base for the attestation of higher level components, i.e. User guest applications. To support attestation of User guest applications, one could instrument an M&A in user-space that is similar in form and function to the M&A described above. The user-space M&A may be chained to the User and Supervisor M&A's to enable complete platform attestations. Furthermore, the implementation is guest OS agile, as the only guest specific components exist entirely within the individual ASPs and the future user-space M&A.

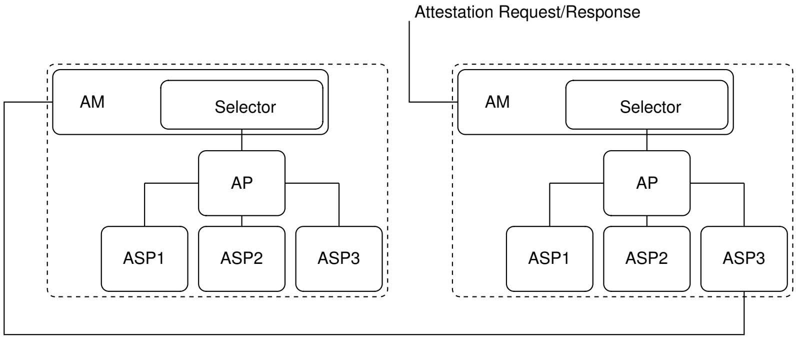


Fig. 3. Composable Measurement and Attestation (M&A) Architecture

Mutual attestation is an important area to consider, and our architecture provides a natural place for this. APs within an M&A may demand attestations as well as providing them, and may use ASPs for verification of the properties asserted by such attestations.

6 Open Problems

Even with our architectural constraints and system design, some aspects of the attestation problem remain difficult to solve. The most difficult principles to satisfy with today's technology are the **Trustworthy Mechanism** and gathering **Comprehensive Information**.

The Trusted Platform Module and related technology from Intel (TxT) [5] and AMD (SVM) [4] are useful means for bootstrapping certain aspects of a self-protecting trust base, but a richer trust base is needed than can be provided by this sort of hardware alone. The emergent hardware technologies only start the problem from a known origin, the core root of trust for measurement, but ultimately the integrity of the trust base depends on the assurance of the "hypervisor" implementation. Specifically required is a means to establish domain separation in order to support a trustworthy mechanism for attestation. Our current implementation uses an off-the-shelf virtualization system – but none of those available today offer the desired balance between flexibility and security. Solutions to this problem might be found either by extending traditional separation kernels or possibly by producing a small, assurance-focused virtual machine hypervisor.

The major problem in gathering comprehensive information is that in order to establish trust in application-level software one first needs to establish trust in the operating system that the software depends on. Today's mainstream operating systems were not designed with assurance or measurement in mind. They are large and complex, containing many dynamic features that make them very difficult to analyze even in the absence of a hostile party. It seems unlikely that this situation will improve until there is either a major shift in the structure of mainstream operating systems or the adoption of a new operating system designed from the beginning with measurement and assurance as a design goal.

7 Existing Approaches to Attestation

There are several early steps toward system attestation in the research community and commercial market today. It is clearly a major component and focus of work being done within the Trusted Computing Group [25] [26] [8], Microsoft [6], and multiple independent researchers [13] [21]. Many of these solutions may act as useful components in a general attestation architecture as described in this paper. Still, none of them fully address this broader notion of attestation or the needs of a flexible architecture.

Trusted Network Connect. Trusted Network Connect (TNC) is a specification from the Trusted Computing Group [26] intended to enable the enforcement of security policy for endpoints connecting to a corporate network.

While Trusted Network Connect is an architecture for attestation, it is of much narrower scope than our approach. Its purpose is to provide trust in endpoints connecting to a network [26], and for this reason it is generally seen as supporting activity at network layers 2 or 3. For this reason, the TNC architecture makes some assumptions about the relationships between parties that make it of limited value for application-level attestations. Once a party has network access, it moves outside the scope of TNC.

In our framework, TNC is best seen not in comparison to our entire architecture but as a special kind of attestation manager. Much of the purpose of the TNC Client (TNCC) is to select the appropriate Integrity Measurement Collectors (IMCs) based on requests from Integrity Measurement Verifiers.

Useful domain separation is not possible in TNC. At load time, each IMC registers what kinds of messages it wishes to receive from the client. If it registers `0xffffffff` then it will receive all messages delivered to all IMCs [27]. Further, it is explicit in the specification that IMCs are loaded into the same memory space as the TNCC, and that a rogue IMC can read and write memory in the TNCC or in other IMCs, misusing credentials, privileges, and message data arbitrarily [27]. Thus, even if the overall TNC process is separated somehow from the target, it is clear that no separation is possible between measurement tools and either the attestation management function or other measurement tools in a system compliant with TNC.

The notion of attestation delegation exists in TNC, but in a very constrained way. The relationships between Policy Enforcement Points and Policy Decision Points is made explicit, making arbitrary delegation difficult at best.

TNC can provide identification of the appraiser to the target, though it is constrained to one very specific identification. Before the integrity measurements are taken, "mutual platform credential authentication" [26] can occur. In the TCG context, this means that the two parties can each verify that the other has a valid unrevoked TPM AIK. However, truly mutual authentication is impossible in TNC due to its nature as a network access protocol. Given that the "server" implicitly already has access, no attestations from the server to the client other than this initial credential exchange is possible. If the client only requires a basic identification then this may be sufficient, but if clients wish to negotiate with servers and proceed differently depending on attested properties, then TNC is unsuitable.

It should be noted that TNC is not a networking or messaging protocol, but rather is intended to be tunneled in existing protocols for managing network access, such as EAP [1].

Due to the asymmetric nature of TNC and the protocols it expects to live within, implementation of complex attestation protocols or nested attestations is unlikely to occur in a way that interoperates with TNC.

Pioneer and BIND. Pioneer and BIND are attestation primitives developed at CMU with very specific design constraints.

BIND [22] is a runtime code attestation service for use in securing distributed systems. It centers around a specific measurement capability which binds a proof of process integrity to data produced by that process. For embedded systems which without flexible attestation needs, BIND may be useful.

Pioneer [21] is an attempt to provide a "first-step toward externally-verifiable code execution on legacy computing systems." Here, legacy means systems with no hardware trust base – Pioneer attempts to solve the attestation problem entirely in software. This faces serious challenges in the presence of a malicious OS, and at least one method is known for an OS to fool Pioneer. Also, the success of Pioneer on any given system requires an immense degree of knowledge about (and control of) the underlying hardware. A trusted third party must know the *exact* model and clock speed of the CPU as well as the memory latency. The system must not be overclocked, must not support symmetric multi-threading, and must not generate system management interrupts during execution of Pioneer. This level of dependency suggests that an attacker with sufficient understanding of the hardware might subvert attestation. In specific, at least one such attack is known in the case of systems with 64-bit extensions. The specific weaknesses referred to are acknowledged by the authors as implementation issues [20].

Another requirement for Pioneer is that the checksum code is the most time-optimal possible code that performs its checksum. No proofs of such optimality exist for any Pioneer-sufficient checksum functions. It remains to be seen if Pioneer can succeed, as newly emerging hardware optimizations will continue to provide attack vectors and make it very difficult to be certain that a given piece of code is time-optimal on all architectures that a user may care about.

Copilot. CoPilot [13] is a system for detecting root kits in a Linux kernel. It periodically computes hashes over key parts of memory that impact kernel execution, compares against a known value, and reports to an external system that enables manual decisions to be made regarding detected changes. CoPilot runs on a PCI add-in card, accessing system memory using DMA. It uses a dedicated port to communicate with the appraiser.

CoPilot does well with respect to some of our principles. It is protected due to existing on separate hardware and via a direct connection to the appraiser. It produces fresh information about a running system.

CoPilot is an advance in attestation technology, but it has limitations. It does not provide a truly comprehensive measurement of the target system because the measurements it produces do not include important information residing in the kernel's dynamic data segment. In addition, since CoPilot does not have direct access to the CPU's registers, it only is able to perform measurements at known memory locations and cannot associate any of its measurements with what is actually running on the processor. Also, the timing of measurement cannot be coordinated with actions on the target. This means that appraisal is difficult as any given measurement might be taken during a moment of inconsistency. Achieving any kind of constrained disclosure is not possible, as there is exactly one appraiser (connected by a cable) and there is no opportunity for the target to participate in the attestation process.

CoPilot is a very specialized kind of attestation system, but it strangely does not take advantage of this. The discussion in Section 4.4 mentioned that *specialists* can be valuable because of their ability to make use of knowledge of the structure of the object being measured. CoPilot, even though it is only used to measure Linux kernels, does not perform structural analysis of data – it only reports hashes of kernel memory. As a result, changes are detected but the meaning of those changes is not feasible to determine. The way that CoPilot's specialized nature is implemented (via dedicated hardware) also means that supporting nested attestation is impossible.

The fact that CoPilot runs on add-in hardware may increase trust in the attestation mechanism and avoid impact on target execution, but at the cost of requiring extra hardware for every target system.

Nexus. Nexus [23] is an effort at Cornell to develop an operating system with particular attention to "active attestation." It enables separation via secure memory regions and moves device drivers into userspace. It introduces "labeling functions," a mechanism for providing dynamic runtime data to appraisers. Measurement tools may be sent to the target system by the appraiser and do not need to be pre-integrated with the base system.

As it involves an entirely new microkernel-based operating system, there are clearly adoption hurdles in the path of Nexus. It is perhaps most useful to think of Nexus not in the role of a guest operating system in our framework, but rather as something one might use for separation purposes instead of a traditional hypervisor. This relationship seems even more relevant in light of the fact that the Nexus project intends to be able to run Linux on top of a Nexus process.

Nexus is not yet released, but one can imagine it playing a part in a variant of our architecture. The ideas of fresh information, comprehensive information, constrained disclosure, and a trustworthy mechanism would clearly resonate with the developers of Nexus. While it does not account for some of the elements in our architecture, it also does not appear to be contradictory with them. As this work emerges into public view it will be worth watching in order to determine how it might be used to satisfy attestation needs.

Property Based Attestation. Our concept of delegated attestation and appraisal is a more general variant of the idea known as *Property-based Attestation*. Jonathan Poritz [18] and Sadeghi and Stüble [19] have each pointed out that the end consumer of an attestation ought to care about security properties of the attester, as opposed to the specific mechanisms employed to achieve those properties. As should be clear from this paper, we strongly agree with this fundamental idea.

Poritz suggests that a solution might include virtualization and trusted third parties, but does not propose a concrete approach. Sadeghi and Stüble go farther, suggesting multiple approaches. Their abstract model of the TCG specifications takes an unusual view of the TPM Seal [25] functionality, which may impact the viability of the proposed solution. Some of their other suggestions for PBA are attractive but require significant changes to the TPM or the TCG software stack.

These authors and several others go on to propose a protocol for performing such property-based attestations [3]. This protocol could be implemented as an AP/ASP combination in our system, as long as some specialist appraiser for it was also implemented.

We go farther than the capabilities shown in work thus far, showing that there can be multiple layers of delegated attestation, that there can be arbitrarily many layers of the platform being appraised, and that the proper appraisers for each of these may be different. The data stored in a hardware TPM is not the only data for which one would wish to delegate the appraisal. Data in virtual TPMs, and even data stored at higher levels in the operating system of a host may be appropriate to delegate to specialists and describe via abstract, property-based attestations.

8 Conclusion

Attestation is an area which will see many technological innovations and developments in the near future. In particular, since the major vendors are introducing improved support for virtualized systems, architectures such as ours should be increasingly easy to implement in a trustworthy way. The semantic explicitness and freshness of the attestations that we propose should allow a common vocabulary across many architectures. Constrained disclosure should encourage systems owners to allow their systems to participate in attestations. Comprehensive information should encourage appraisers to place credence in well-supported claims, particularly given underlying trustworthy attestation mechanisms. We have attempted to clarify the way that existing work can be used to contribute to our goals.

References

1. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard) (June 2004)
2. Balacheff, B., Chen, L., Pearson, S., Plaquin, D., Proudler, G. (eds.): Trusted Computing Platforms: TCPA Technology in Context. Prentice Hall PTR, Upper Saddle River (2003)

3. Chen, L., Landfermann, R., Löhr, H., Rohe, M., Sadeghi, A.-R., Stüble, C.: A protocol for property-based attestation. In: STC 2006: Proceedings, First ACM Workshop on Scalable Trusted Computing, pp. 7–16. ACM Press, New York (2006)
4. AMD Corporation. Amd64 architecture programmer's manual volume 2: System programming rev 3.11 (January 2006), http://www.amd.com/usen/assets/content_type/white_papers_and_tech_docs/24593.pdf
5. Intel Corporation. Intel trusted execution technology (November 2006), http://download.intel.com/technology/security/downloads/31516803.pdf
6. Microsoft Corporation. Ngscb official page (2007), http://www.microsoft.com/resources/ngscb/default.mspx
7. Grawrock, D.: The Intel Safer Computing Initiative. Intel Press (2006)
8. TCG Best Practices Group. Design, Implementation, and Usage Principles for TPM-Based Platforms. Version 1.0 (May 2005)
9. Guttman, J.D.: Authentication tests and disjoint encryption: a design method for security protocols. Journal of Computer Security 12(3/4), 409–433 (2004)
10. Guttman, J.D., Herzog, J.C., Ramsdell, J.D., Sniffen, B.T.: Programming cryptographic protocols. In: De Nicola, R., Sangiorgi, D. (eds.) TGC 2005. LNCS, vol. 3705, pp. 116–145. Springer, Heidelberg (2005)
11. Guttman, J.D., Thayer, F.J., Carlson, J.A., Herzog, J.C., Ramsdell, J.D., Sniffen, B.T.: Trust management in strand spaces: A rely-guarantee method. In: Schmidt, D. (ed.) ESOP 2004. LNCS, vol. 2986, pp. 325–339. Springer, Heidelberg (2004)
12. Haldar, V., Chandra, D., Franz, M.: Semantic remote attestation – a virtual machine directed approach to trusted computing. In: Proceedings of the Third virtual Machine Research and Technology Symposium, May 2004, pp. 29–41. USENIX (2004)
13. Petroni Jr., N.L., Fraser, T., Molina, J., Arbaugh, W.A.: Copilot - a coprocessor-based kernel runtime integrity monitor. In: USENIX Security Symposium, pp. 179–194. USENIX (2004)
14. Katsuno, Y., Watanabe, Y., Yoshihama, S., Mishina, T., Kudoh, M.: Layering negotiations for flexible attestation. In: STC 2006: Proceedings, First ACM Workshop on Scalable Trusted Computing, pp. 17–20. ACM Press, New York (2006)
15. Kerber, R.: Advanced tactic targeted grocer: 'Malware' stole Hannaford data. The Boston Globe p. 1, 18 March (2008)
16. Loscocco, P., Smalley, S.: Integrating flexible support for security policies into the linux operating system. Technical report, NSA, NAI Labs (April 2001)
17. Millen, J., Guttman, J., Ramsdell, J., Sheehy, J., Sniffen, B.: Call by contract for cryptographic protocol. In: FCS-ARSPA (2006)
18. Poritz, J.A.: Trust[ed — in] computing, signed code and the heat death of the internet. In: SAC 2006: Proceedings of the 2006 ACM symposium on Applied computing, pp. 1855–1859. ACM Press, New York (2006)
19. Sadeghi, A.-R., Stüble, C.: Property-based attestation for computing platforms: caring about properties, not mechanisms. In: NSPW 2004: Proceedings, 2004 Workshop on New Security Paradigms, pp. 67–77. ACM Press, New York (2004)
20. Seshadri, A.: Pioneer web page, http://www.cs.cmu.edu/
21. Seshadri, A., Luk, M., Shi, E., Perrig, A., van Doorn, L., Khosla, P.: Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms. In: Proceedings of ACM Symposium on Operating Systems Principles (SOSP), October 2005, pp. 1–16 (2005)

22. Shi, E., Perrig, A., Van Doorn, L.: BIND: A time-of-use attestation service for secure distributed systems. In: Proceedings of IEEE Symposium on Security and Privacy (May 2005)
23. Shieh, A., Williams, D., Sirer, E.G., Schneider, F.B.: Nexus: a new operating system for trustworthy computing. In: SOSP 2005: Proceedings of the twentieth ACM symposium on Operating systems principles, pp. 1–9. ACM Press, New York (2005)
24. Stone, B.: 11 charged in theft of 41 million card numbers. The New York Times, p. B 1, 5 August (2008)
25. Trusted Computing Group. TPM Main Specification, version 1.1b edition (2001), https://www.trustedcomputinggroup.org/downloads/tcg_spec_1_1b.zip
26. Trusted Computing Group. TCG Trusted Network Connect: TNC Architecture for Interoperability. Version 1.1 (May 2006)
27. Trusted Computing Group. TCG Trusted Network Connect TNC IF-IMC, Version 1.1 (May 2006)

A Novel Solution for End-to-End Integrity Protection in Signed PGP Mail

Lijun Liao and Jörg Schwenk

Horst Görtz Institute of IT-Security,
Ruhr-University Bochum, Germany
{lijun.liao,joerg.schwenk}@nds.rub.de

Abstract. PGP mail has been widely used to provide the end-to-end authentication, integrity and non-repudiation. However it has the significant drawback that the email header is unauthentic. DKIM protects specified header fields, but only between the sending server and the receiver. These lead to possible impersonation attacks and profiling of the email communication, and encourage spam and phishing activities. In this paper we propose an approach to extend PGP mail to support end-to-end integrity of whole email, namely the whole content and selected header fields. This approach is fully compatible with PGP mail. Under some reasonable assumption our approach can help to reduce spam efficiently.

1 Introduction

Emails are not protected as they move across the Internet. Often information being transmitted is valuable and sensitive such that effective protection mechanisms are desirable in order to prevent information from being manipulated or to protect confidential information from being revealed by unauthorized parties. A large number of email security mechanisms have been meanwhile developed and standardized, which build a solid basis for secure email communication. The most well-known and widely employed mechanisms are S/MIME and (Open)PGP mail. PGP and OpenPGP are two similar formats. The main difference is that OpenPGP does not use the patent algorithms which are contained in PGP. For convenience, we use in the following only PGP for both PGP and OpenPGP.

Based on the analysis of the related work in Section 2 we make clear that further improvement is needed. In this paper we discuss how to extend PGP mail to support header protection with compatibility with prior versions. This can be proven by our prototype implementation. We discuss also how to employ our approach to reduce spam in emails.

The rest of this paper is organized as follows. We review the related work in Section 2. The signature in PGP format is briefly described in Section 3. In Section 4 we list the goals of our approach which is focused in Section 5. We analyze our approach in Section 6; Before we conclude our paper in Section 8, we describe briefly our prototype implementation in Section 7.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 19–32, 2008.

2 Related Work

End-to-End Security Mechanisms: PGP mail [1,2] is one of the most widely propagated mechanisms to provide authentication, message integrity, non-repudiation of origin, and data confidentiality. The email sender signs the message body using his private key. The receiver verifies the signature with the corresponding public key after receiving signed message. However, in PGP mail, only the body of the email message is protected. Most header fields, such as `To`, `Date`, and `Subject`, are remain unprotected, and the `From` and `Sender` are only secure if the receiver checks that the address in the `From` or `Sender` of a mail message matches the Internet mail address in the signer ID of the signer's certificate. The most email clients[1] do not check it. Fig. 1 shows that the Thunderbird (with PGP plug-in Enigmail [3]) cannot detect the modification of any email header fields[2].

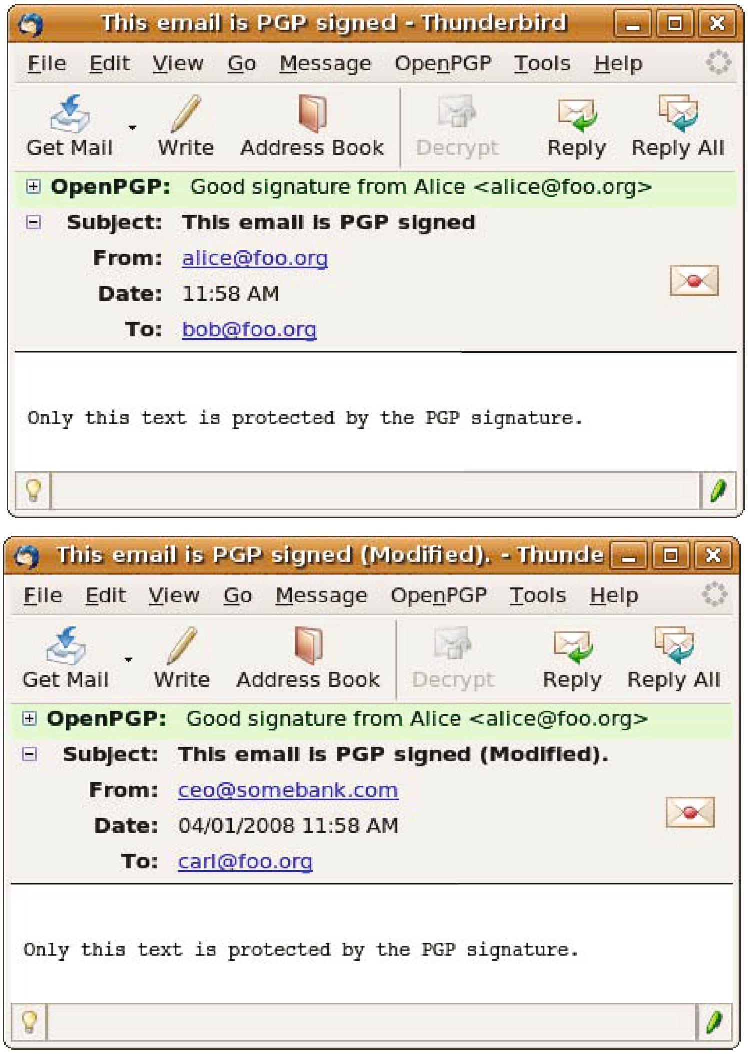
In the following we analyze some approaches that provide sender authentication or the authentication of some email header fields: S/MIME 3.1, Sender ID, SPF, DKIM, and LES. They are designed to use X.509 based techniques. With light modification, they can be deployed with PGP based techniques, e.g. using PGP certificate instead of X.509 certificate, and using web of trust instead of public key infrastructure (PKI).

To send a signed email in S/MIME 3.1, one prepares an email m_1 as usual. A new email m_2 is then created with the same header field as $m1$, a media block with $m1$ as its content is then added to m_2. Now a signature is added to m_2 to protect the content of m_2, namely the media block with m_1. In this case, the content and header of m_1 are both protected by the signature. However, this approach is associated with the following disadvantages:

1. All header fields in m_1 must also appear in m_2 (i.e., they must be presented doubly) so that the email is conform to RFC 2822 [4] and the MUAs and MTAs know how to send the email.
2. Only header in m_1 is protected, but not the one in m_2. As stated in [5], it is up to the receiving client to decide how to present the protected fields in m_1 along with the unprotected ones in m_2. Since there are no well-known clients that support S/MIME 3.1, we cannot test S/MIME 3.1 in praxis. Without loss of the generality, we assume the behaviour of future clients as follows: the following header fields, if present, are shown in most clients: `From`, `Sender`, `To`, `CC`, `Date`, `Subject`, and `Reply-To`. If the same header field is present in m_1 and m_2, only the one in m_1 is presented. If a header field is only presented in m_2, it will be also shown. Most emails do not contain the header fields `Sender` and `CC`, hence one can add these header fields in the outer header to confuse the receivers.

[1] Most email clients do not support PGP mail directly, but with the help of additional plug-ins. For convenience, we refer in this paper the email clients to the clients with the PGP plug-ins.

[2] Enigmail does not even check whether the email sender matches the Internet address in the PGP certificate.


This email is PGP signed - Thunderbird
File Edit View Go Message OpenPGP Tools Help
Get Mail
Write
Address Book
Decrypt
Reply
Reply All

OpenPGP: Good signature from Alice <alice@foo.org>

Subject: This email is PGP signed
From: alice@foo.org
Date: 11:58 AM
To: bob@foo.org

```
Only this text is protected by the PGP signature.
```

This email is PGP signed (Modified). - Thunde
File Edit View Go Message OpenPGP Tools Help
Get Mail
Write
Address Book
Decrypt
Reply
Reply All

OpenPGP: Good signature from Alice <alice@foo.org>

Subject: This email is PGP signed (Modified).
From: ceo@somebank.com
Date: 04/01/2008 11:58 AM
To: carl@foo.org

```
Only this text is protected by the PGP signature.
```

Fig. 1. A signed PGP mail in Thunderbird 2.0 with PGP plug-in Enigmail. The original message (top) is sent from `alice@foo.org` to `bob@foo.org`. In the modified message (bottom), the `From` field is changed to `ceo@somebank.com`, the fields `To`, `Date`, and `Subject` are also modified; however, the signature is still treated as valid in Thunderbird.

3. It complicates the receiver to show the email. It is difficult to determine whether the message within the `message/rfc822` wrapper is the top-level message (as in S/MIME 3.1), or the complete `message/rfc822` MIME entity is another encapsulated mail message.

Sender Verification Frameworks: There are several path based sender verification frameworks for email, e.g. Sender ID [6], and SPF [7]. Such frameworks protect only the direct sending server and are not suitable for email messages which are forwarded by other sending servers.

DKIM [8] protects the important header fields using digital signatures. The sending server signs each outgoing email, including some email header fields and the body. The public key used to verify the signature is placed in the sending server. Such approaches can protect more header information; however, the communication between the sender and sending server remains unprotected. Additionally the sending and receiving servers are vulnerable to Denial-of-Service (DoS) attacks. An attacker may flood the sending server with million emails and force the sending server to sign them. Such an attack can similarly be applied to the receiving server by sending million emails with valid signature formats (the signature may be invalid, e.g. a random number as the signature value).

LES [9] is an extension of DKIM and allows the sender to sign the header fields and body using DKIM signature. It seems to provide end-to-end authentication, message integrity and non-repudiation. However, the private key is generated by some server and is sent unprotected to the sender via email; hence at least one other than the signer (sender) knows the private key. Therefore no real end-to-end security is achieved. Since the receiving server verifies the signatures of all incoming emails, it is also vulnerable to DoS attacks.

Additionally, all approaches above are vulnerable to DNS spoofing attacks, and one can still send spam if he has the legal email address, which can be gotten very easily.

Spam: MAAWG estimates that 74–81% of incoming emails between October 2005 and June 2006 were spam [10]. There are a number of methods in use to manage the volume and nature of spam. Many organizations employ filtering technology. However, the emails today do not contain enough reliable information to enable filters and recipients to consistently decide if messages are legitimate or forged. Others use publicly available information about potential sources of spam. These policy and filter technology measures can be effective under certain conditions, but over time, their effectiveness degrades due to increasingly innovative spammer tactics.

With our approach we enable a sender to provide proof that an email is legitimate and not from a spammer; and more effective spam control mechanisms can be built to reduce both the amount of spam delivered and the amount of legitimate emails that are blocked in error. Although the application of our extension for anti-spam is based on an assumption (see Assumption 1) that is not satisfied today, we believe that this goal can be achievable shortly.

3 Signature in PGP Format

PGP can be used to digitally sign, digest, authenticate, or encrypt arbitrary message content. PGP signature is carried by a signature packet which contains the hash algorithm, public key algorithm, hashed subpacket data, unhashed subpacket data, and the signature value. The subpacket data may contain some data related to the signature. If the subpackets which specify the the signature creation time, the key ID of the key issuing the signature, are present, they must be contained in the hashed subpacket data.

A hash value is computed over the content to be signed, the hash algorithm, the public key algorithm, and the hashed subpacket data. In a email with PGP signature, the content to be signed is the content of email. The signature value is then computed over the hash value with signer's private key.

PGP defines a subpacket type `notation` for extension. There may be more than one `notation` subpacket in a signature. By inserting a `notation` subpacket to the hashed subpacket data we extend PGP mail for header protection as shown in Section 5.2.

4 Goals of Our Approach

Based on the analysis in Section 2 we propose an approach with the following advantages:

1. *End-to-end security of the complete message*: Authentication, integrity and non-repudiation should be achieved for not only the email body, but also some important header fields.
2. *Compatibility with prior versions*: Old clients that are only PGP mail capable should not treat signatures in emails with our approach as invalid.
3. *Simple implementation of clients*: It should be easy to implement our approach.
4. *Support for anti-spam*: Our approach should help to reduce spam.

5 Extension in PGP Mail

In this section we present an approach to achieve the goals described in Section 4. The basic concept is to specify the header fields that should be signed, the hash algorithm, and hash value computed over the specified header fields. Such information is contained in a hashed subpacket of type `notation` in PGP so that it is protected by the signature.

5.1 Header Protection Entity

We use similar format as in DKIM to identify the headers that are signed, namely a colon-separated list of header names. For example, the field name list `From:To:Date:Subject:Reply-To` indicates that the first header fields (from bottom to top) `From`, `To`, `Date`, `Subject`, and `Reply-To` are signed. If a referenced header field does not exist, an empty string is used instead. This is useful to prevent adding undesired header fields. For an email without the field `CC` field while creating the signature, we can use $\cdots$`:CC` (where $\cdots$ is for list of any other field names) to avoid the insertion of a new CC header. The signer can sign multiple instances of a header by including the header name multiple times; such field instances are then signed in order from the bottom of the header block to the top. If there are n instances of a header, including the header name $n + 1$ times avoids the insertion of a new instance. Considering the example in Fig. 2, $\cdots$:To:To avoids adding a second field To.

To prevent from adding an additional field or a new one, the corresponding field name should be listed at least once more than the expected times of its appearance. In general, the fields From, Sender, To, CC, Date, Subject, and Reply-To are important, and each of them is only allowed once in the email. Hence, the following field name list From:From:Sender:Sender:To:To:CC:CC: Date:Date:Subject:Subject:Reply-To:Reply-To, denoted as fnl, should be used at least.

```
From: Alice <alice@foo.org>
TO: Bob <bob@foo.org>
Subject : Test
Date: Tue, 6 Mar 2007 09:21:36 +0100
```

Fig. 2. Header block of an email. The name of field To is in uppercase.

```
From: Alice <alice2@foo.org>
To: Carl <carl@foo.org>
TO: Bob <bob@foo.org>
CC: Eva <eva@foo.org>
Subject : Test (modified)
Date: Tue, 6 Mar 2007 09:21:36 +0100
```

Fig. 3. Modified header block of an email from Fig. 2. The field Subject is modified, and a second To and a new Cc are added.

We consider now the examples in Figures 2 and 3. The result of applying fnl to the header blocks are depicted in Fig. 4 and Fig. 5, respectively. The modification can be detected clearly by comparing both results.

Email, specially the email header fields, may be modified by some mail servers and relay systems. Some signers may demand that any modification of email headers result in a signature failure, while some other signers may accept modification of headers within the bounds of email standards, e.g. the addition or deletion of white spaces, and changing the lowercase or uppercase of the letters of field name. For these requirements we use the header canonicalization algorithms simple and relaxed defined in the DKIM specification [8, §3.4.1, §3.4.2] respectively.

```
From: Alice <alice@foo.org>
TO: Bob <bob@foo.org>
Date: Tue, 6 Mar 2007 09:21:36 +0100
Subject : Test
```

Fig. 4. Result of applying `From:From:Sender:Sender:To:To:CC:CC:Date:Date:Subject:Subject:Reply-To:Reply-To` to header block in Fig. 2

```
From: Alice <alice2@foo.org>
TO: Bob <bob@foo.org>
To: Carl <carl@foo.org>
CC: Eva <eva@foo.org>
Date: Tue, 6 Mar 2007 09:21:36 +0100
Subject : Test (modified)
```

Fig. 5. Result of applying `From:From:Sender:Sender:To:To:CC:CC:Date:Date:Subject:Subject:Reply-To:Reply-To` to modified header block in Fig. 3

The `simple` header canonicalization algorithm does not change headers in any way; hence any modification of the headers, e.g. adding a space in one header or change the field name "TO" to "To", will invalidate the signature.

The `relaxed` header canonicalization algorithm canonicalizes the headers in order as following:

1. convert all field names to lower case;
2. unfold all field continuation lines as described in RFC 2822 [4];
3. convert all sequences of one or more white spaces, e.g. space or tab, to a single space;
4. delete all white spaces at the end of each unfolded field value;
5. delete any white spaces remaining before and after the colon which separates the field name from the field value.

Hence, `relaxed` allows the following changes:

1. Changing the uppercase or lowercase of each character in field name.
2. Adding or removing white spaces between field name and field value
3. Changing the number of continuing white spaces (at least 1) in a field value
4. Replacing new line in field value with white spaces or vice versa. Note that the new line must start with at least one white space so that it belongs to the same field.

By applying the canonicalization method `relaxed` to the selected header fields in Fig. 4 and Fig. 5, we get the results in Fig. 6 and Fig. 7, respectively.

To protect the intended fields, the simplest method is to put the canonicalized result together with the field name list and the canonicalization method in a

```
from:Alice <alice@foo.org>
to:Bob <bob@foo.org>
date:Tue, 6 Mar 2007 09:21:36 +0100
subject:Test
```

Fig. 6. Result of applying the canonicalization method `relaxed` to the selected header block in Fig. 4

```
from:Alice <alice2@foo.org>
to:Bob <bob@foo.org>
to:Carl <carl@foo.org>
cc:Eva <eva@foo.org>
date:Tue, 6 Mar 2007 09:21:36 +0100
subject:Test (modified)
```

Fig. 7. Result of applying the canonicalization method `relaxed` to the selected header block in Fig. 5

hashed subpacket of type `notation`. Its main disadvantage is that the signature size will be increased enormously, and is proportional linearly to the canonicalized fields, since the signature size without this attribute is almost fixed and small.

A better method is to compute the hash value over the canonicalized result. The hash value and the hash algorithm, instead of the canonicalized result itself, are then saved. In this case, the signature size is much less than the method above, since it is proportional linearly to the size of field name list instead of the canonicalized fields.

As described above, the former needs much more size than the latter. Hence we consider in this paper only the latter.

We introduce some abstract notations to simplify the explication. We denote $\mathit{\Gamma}(l{:}\textbf{string}, c{:}\textbf{string}, \alpha{:}\textbf{string}, \gamma{:}\textbf{bytes})$ as a header protection entity, where l is list of header field names, c is header canonicalization method, α is hash algorithm, and γ is hash value, e.g. $\mathit{\Gamma}($`'From:To:Date:Subject'`, `'relaxed'`, `'SHA1'`,$\gamma_{SHA1})$, where γ_{SHA1} has 20 bytes. Let $\mathcal{E}= (L, C, H)$ be the notation for the creation and verification of $\mathit{\Gamma}(l, c, \alpha, \gamma)$ for the email header block φ. Note that the processes to verfiy and create the signature over the email body remain unchanged. The L algorithm retrieves the header fields specified by the list l from φ and is denoted as $L_l(\varphi)$. The C algorithm is the canonicalization algorithm. $C_c(\tau)$ canonicalizes email header τ with the canonicalization method c. H is the secure one-way hash function, and $H_\alpha(m)$ denotes the computation of hash value over message m with algorithm α. The validity of a header protection entity $\mathit{\Gamma}$ for the email header block φ is defined in Definition 1.

Definition 1. *An entity $\mathit{\Gamma}(l, c, \alpha, \gamma)$ is valid for the email header block φ if and only if $\gamma = H_\alpha(m)$, where $m = C_c(\tau), \tau = L_l(\varphi)$.*

5.2 Extension in PGP Mail

As described in Section 3, the email content, hash algorithm, public key algorithm (the algorithm to sign the hash value), and hashed subpackets are protected by the signature. To make our header protection entity protected by the signature to prevent from modifying, we use a hashed subpacket `notation` to represent the entity $\Gamma(l, c, \alpha, \gamma)$. A subpacket of type `notation` is defined in [2, §5.2.3.15] as follows:

```
notation ::= {
  4 octets of flags,
  2 octets of name length (M),
  2 octets of value length (N),
  M octets of name data (Name),
  N octets of value data (Value) }
```

A `notation` subpacket has a name and a value. The `flags` field holds four octets of flags, for our extension we set all octets to zero. The `Name` field carries the name of this `notation`, and `Value` carries the value data specified by `Name`. We use `RFC2822HeaderProtect@rub.de` as `Name`; hence $M = length(Name) = 27 = 0x001B$, where $length(x)$ returns the number of octets contained in x. The entity $\Gamma(l, c, \alpha, \gamma)$ is specified by an object of type `RFC28222HeaderProtectType`.

```
RFC2822HeaderProtectType ::= {
 1 octet canon. algorithm (CanonAlgo),
 1 octet digest algorithm (DigestAlgo),
 2 octets field name list length (K),
 K octets field name list (FieldNameList),
 1 octet hash value length (L),
 L octets hash value (HashValue) }
```

The `CanonAlgo` field carries the header canonicalization algorithm c: 1 for `simple` and 2 for `relaxed`. The `DigestAlgo` field carries the digest algorithm α as defined in [2, §9.4], e.g. 2 for SHA1. The `FieldNameList` field specifies the list of header names l of K octets, and the hash value γ of L octets is specified in the `HashValue` field.

Some systems, such as GnuPG, allow only ascii-text in notation value, but not arbitary-text. The object of type `RFC28222HeaderProtectType` should be first encoded in such systems. As a generic solution the first byte of the notation value indicates the encoding algorithm. The rest bytes are then the encoded object of type `RFC28222HeaderProtectType`. The predefined algorithms are: 'a' for armor encoding, 'b' for base64 encoding, and 'n' for no encoding.

Figures 8 and 9 depicts the `notation` subpacket with the following contents:

- the canonicalization algorithm is `relaxed`,
- the digest algorithm is `SHA1`, and
- the header fields under the field name list `From:From:Sender:Sender:To:To:CC:CC:Date:Date:Subject:Subject:Reply-To:Reply-To` are protected from modification.

```
Offset  Length                        Content
   0       4:        Flags { 0 }
   4       2:       M { 27 }
   6       2:       N { 105 }
   8      27:       Name { "RFC2822HeaderProtect@rub.de" }
  35     105:       Value {
  35       1:          EncodingAlgorithm {'n' = no encoding}
  36       1:          CanonAlgo { 2 = relaxed }
  37       1:          DigestAlgo { 2 = SHA1 }
  38       2:          K { 79 }
  40      79:          Reference { "From:From:Sender:Sender:To:To:
                                    CC:CC:Subject:Subject:Date:Date:
                                    Reply-To:Reply-To" }
  119      1:          L { 20 }
  120     20:          HashValue { DD 6F 37 34 BE 9B 43 DA F4 B8
                                   17 5B 99 EA 10 64 9B 78 69 2D }
  140      -:       }
```

Fig. 8. A notation subpacket which specifies the header protect entity

```
Offset  Length                        Content
   0       4:        Flags { 0 }
   4       2:       M { 27 }
   6       2:       N { 142 }
   8      27:       Name { "RFC2822HeaderProtect@rub.de" }
  35     142:       Value {
  35       1:           EncodingAlgorithm {'b' = base64 encoding}
  36     141:           Encoded Header Protection Entity{
                            AgIAT0Zyb206RnJvbTpTZW5kZXI6U2VuZGVyOl
                            RvOlRvOkNDOkNDOlN1YmplY3Q6U3ViamVjdDp
                            EYXRlOkRhdGU6UmVwbHktVG86UmVwbHktVG
                            8Uci0rT7/2p+Jm6P/jjDrX7cb4Fpk=
                        }
  177      -:     }
```

Fig. 9. A notation subpacket which specifies the base64 encoded header protect entity

In Figure 8 the entity is not encoded, while it is base64 encoded in Figure 9.

To send a signed PGP mail with our extension, the sending client behaves as usual with the following exception: before it signs, it generates a `notation` with the name `RFC2822HeaderProtect@rub.de` to specify the entity $\Gamma(l, c, \alpha, \gamma)$ for the email φ within the signed subpacket data.

Assume that the receiving client receives an email of header block φ' with the `notation` that specifies $\Gamma(l', c', \alpha', \gamma')$. If the client implements our approach, it does the following:

1. retrieve the header fields referenced by `FieldNameList` l': $\tau' = L_{l'}(\varphi')$;
2. canonize $\tilde{\tau}$ with the canonicalization algorithm c': $m' = C_{c'}(\tau')$;
3. compute the hash value over m': $\tilde{\gamma} = H_\alpha(m')$;
4. compare $\tilde{\gamma}$ with γ' specified in `digest`. If $\tilde{\gamma} \neq \gamma'$, terminate the verification process and consider the signature as invalid, otherwise do other checks as usual.

Since the most clients ignore unrecognized notations, they are still able to verify the signature as usual, even when they do not implement our approach. However, the header modification can no more be detected.

6 Analysis

This section shows how the goals mentioned in Section 4 can be satisfied in our approach.

End-to-end security of the complete message: The authentication, integrity and non-repudiation of the email body are achieved by the basic PGP mail mechanism, and the ones of the important header fields are achieved by the `notation` with the name `RFC2822HeaderProtect@rub.de` within the hashed subpackets in PGP signature.

Compatibility with prior versions: In PGP mail the receiving client that does not implement our approach will ignore the unrecognized notations respectively; hence the signature can be verified as usual.

Simple implementation of clients: A PGP mail capable client should be extended as follows to support the header protection:

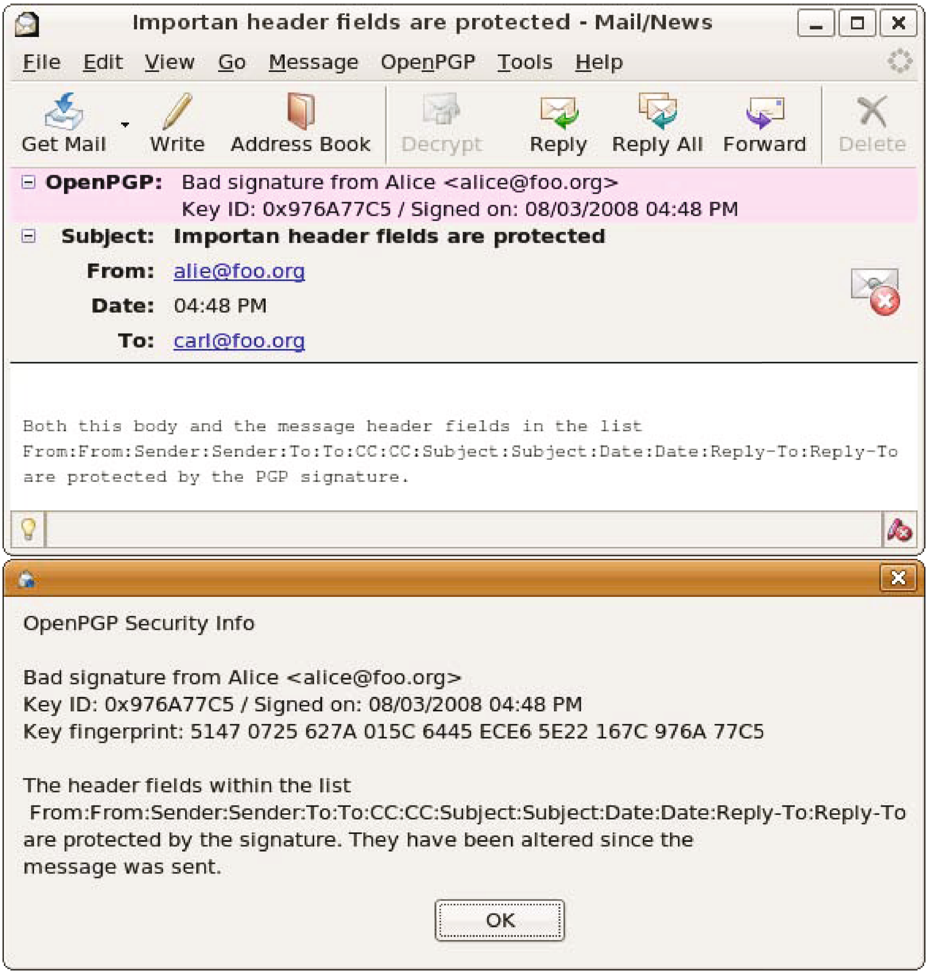
- Generating/verifying the header protection entity.
- Adding/retrieving the header protection entity to/from the hashed packets.
- Informing the users the result of the header protection verification: which header fields are protected and whether they are modified.

The existing PGP capable clients can be further used and need only to be extended. The most complex function for this extension is the hash computation, which is implemented in all PGP capable clients. Hence we can just call the existed one. All other functions should be not difficult to implement.

Support for anti-spam: After researching some spam archives, e.g. [11,12], we argue that most spam messages are unsigned. Therefore most of spam can be rejected if only signed emails (with valid signatures) are accepted. The email header is not protected by PGP mail; hence a clever spammer is able to send signed spam and modify the header. To provide more efficient and proper mechanism against spam, signed emails with our extension should be applied under Assumption 1.

Assumption 1. *An email system should satisfy the following conditions:*

1. *The users have trusted PGP certificates and send only signed emails with our approach.*
2. *At least the header fields* `From`, `Sender`, `To`, `CC`, `Date`, `Subject`, *and* `Reply-To` *must be signed to prevent from modification of the existing fields and addition of new fields in the above list.*
3. *The signer is identified by the address either in the* `Sender` *field, or in the* `From` *field if there is no* `Sender` *field. It must match the address in the* `signerId` *of the signing certificate;*
4. *Each email has limited receivers in the headers* `To` *and* `CC`*;*



OpenPGP: Bad signature from Alice <alice@foo.org>
Key ID: 0x976A77C5 / Signed on: 08/03/2008 04:48 PM

Subject: Importan header fields are protected

From: alie@foo.org

Date: 04:48 PM

To: carl@foo.org

```
Both this body and the message header fields in the list
From:From:Sender:Sender:To:To:CC:CC:Subject:Subject:Date:Date:Reply-To:Reply-To
are protected by the PGP signature.
```

OpenPGP Security Info

Bad signature from Alice <alice@foo.org>
Key ID: 0x976A77C5 / Signed on: 08/03/2008 04:48 PM
Key fingerprint: 5147 0725 627A 015C 6445 ECE6 5E22 167C 976A 77C5

The header fields within the list
From:From:Sender:Sender:To:To:CC:CC:Subject:Subject:Date:Date:Reply-To:Reply-To
are protected by the signature. They have been altered since the
message was sent.

Fig. 10. The header field `To` is modified from `bob@foo.org` to `carl@foo.org`, and this modification is detected by extended Enigmail

5. *An email is accepted if and only if it has valid signature (i.e. the email is not modified and signed by a person with trusted certificate) and the receiver either is directly contained or is a member of the mailing list contained in the header* `To` *or* `CC`*;*
6. *The verification is processed by the email client, not the receiving server to avoid the DoS attack.*

Even if a spammer has a trusted certificate, he cannot sign the email once and send it to million victims. Assuming that max. 10 receivers are allowed in an email, if the spammer wishes to send a spam to 1,000,000 victims, he must sign at least 100,000 times which takes much time and cost. In fact, the spammer put only one receiver in the `To` field to confuse the victim that he is the only intended receiver; hence the spammer must sign the email individually for each victim which requires much more time and cost. Without our extension, the header can not be signed; therefore the spammer needs to sign the email only once, and replaces the receivers (in header fields `To` and `CC`) without invalidating the signature.

Our extension can help reduce spam, but will not stop spam entirely. It should be used together with other technologies, such as filtering and policy technologies.

7 Prototype Implementation

We have extended the OpenPGP Plug-In Enigmail for Thunderbird to support our approach. We have created signed PGP mail messages and then modified some header fields. This modification could only detected in the extended Enigmail. An example with a modified `To` field is given in Fig. 10.

8 Conclusion and Future Work

In this paper we discussed how to extend PGP mail to provide the end-to-end protection of the email header fields. Our approach does not invalidate the signature even if the receiving client does not understand it. The existing clients can be simply extended to support it. With some reasonable assumptions, our approach provides efficient method to struggle with the spam. Since PGP mail is widely accepted, header protection implemented here may have great impact.

As our future work, we will extend the current approach to implement extensions for the popular email clients to support our approach (extension of Enigmail is partly finished), suggest the web mail providers to support PGP mail with our approach in their web mail interfaces.

References

1. Elkins, M., Torto, D.D., Levien, R., Roessler, T.: MIME Security with OpenPGP, IETF RFC 3156 (August 2001)
2. Callas, J., Donnerhacke, L., Finney, H., Thayer, R.: OpenPGP Message Format, IETF RFC 2440 (November 1998)

3. The enigmail project - a simple interface for openpgp email security, http://enigmail.mozdev.org
4. Resnick, P.: Internet Message Format, IETF RFC 2822 (April 2001)
5. Ramsdell, B. (ed.): Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1, IETF RFC 3851 (July 2004)
6. Lyon, J., Wong, M.: Sender ID: Authenticating E-Mail, IETF RFC 4406 (April 2006)
7. Wong, M., Schlitt, W.: Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1, IETF RFC 4408 (April 2006)
8. Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., Thomas, M.: DomainKeys Identified Mail (DKIM) Signatures, IETF RFC 4871 (May 2007), http://www.ietf.org/rfc/rfc4871.txt
9. Adida, B., Chau, D., Hohenberger, S., Rivest, R.L.: Lightweight email signatures (February 2006), http://theory.lcs.mit.edu/rivest/AdidaChauHohenbergerRivest-LightweightEmailSignatures.pdf
10. Email metrics program: The network operators' perspectivereport #3 - 2nd quarter 2006, Messaging Anti-Abuse Working Group(MAAWG), Tech. Rep., (November 2006), http://www.maawg.org/about/FINAL_2Q2006_Metrics_Report.pdf
11. Cormack, G.V., Lynam, T.R.: TREC 2005 spam track public corpora. (2005), http://plg.uwaterloo.ca/gvcormac/treccorpus/
12. Cormack, G.V., Lynam, T.R.: TREC 2006 spam track public corpora. (2006), http://plg.uwaterloo.ca/gvcormac/treccorpus06/

Unclonable Lightweight Authentication Scheme[⋆]

Ghaith Hammouri, Erdinç Öztürk, Berk Birand, and Berk Sunar

Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609-2280
{hammouri,erdinc,bbirand,sunar}@wpi.edu

Abstract. We propose a lightweight, tamper-resilient challenge-response authentication scheme. The scheme that we propose (HB+PUF) is a variant of the PUF-HB protocol [1] which utilizes Physically Unclonable Functions (PUFs). We reduce the security of (HB+PUF) in the active attacker model to solving the LPN problem. The proposed scheme enjoys strong tamper-resilience due to the PUF properties. We present a proof of concept implementation for the proposed protocol. To generate the random bits needed for the protocol, we reuse the PUF circuit as a Random Number Generator (RNG). This construction shows to be cost-effective since we will be using the same hardware for authentication as well as random number generation without incuring any significant overhead. The overall scheme including storage, tamper-resilience and RNG can be achieved with less than 1000 gates. The small footprint should be ideal for constrained environments such as RFID's, smart cards, and sensor networks.

Keywords: Provable security, tamper-resilience, lightweight, random number generation, PUF, HB+.

1 Introduction

Lightweight cryptography has been receiving more attention in the past few years [10,21]. This attention is mainly motivated by the boom in the next generation ubiquitous networks. With highly constrained devices such as wireless sensor nodes, RF identification devices (RFIDs), and smartcards as their main building block, these networks pose an urgent need for affordable cryptography. A number of the known results so far reduce the size of the hardware by serializing classical cryptographic protocols, thus decreasing the circuit's footprint [11,22,37,28]. For example, in [37] and [28] the authors present a lightweight implementation of DESL, a variant of DES. The presented function uses a single S-box 8 times, rather than using the 8 S-boxes needed for DES. With this reduction, the authors manage to serialize the implementation, therefore decreasing the footprint. Although these results are very exciting, the approach itself seems to be inherently limited. Most classical cryptographic protocols were designed

[⋆] This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and CNS-0716306.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 33–48, 2008.

with little attention to the hardware implementation. Therefore, the protocol may not lend itself to serialization. Even from a broader perspective, the notion of taking classical protocols and attempting to squeeze them into a smaller circuit seems to eventually face natural limitations. A different yet exciting approach is to explore new protocols which are motivated by the hardware and which are lightweight in nature [6,31,17,41]. This approach seems harder due to the typical computational demands of cryptography. Nevertheless, it also seems more likely to provide a fundamental solution. This point of view becomes stronger when we consider the diversity of the attacks that target the hardware. While typical cryptographic protocols might be very secure in theory, one has to take into account attacks which exploit so called side-channels [26,25]. These attacks are roughly 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 [27]. Not surprisingly, active attacks are more powerful and are more difficult to prevent. A tamper-resilient hardware could help in securing devices from both passive and active side-channel attacks. With all this in mind, it becomes vital to introduce secure lightweight cryptographic protocols which are motivated by the hardware, and which are tamper-resilient.

Two promising candidates for such a task are Physically Unclonable Functions (PUFs), and the HB-based authentication family. A PUF is a physical pseudorandom function which exploits the small variances in the wire and gate delays inside an integrated circuit (IC). Even when the ICs are designed to be logically identical, the sensitive delay variances will be unique to each IC. These properties will result in providing the hardware with a level of tamper-resilience. However, these devices are not known to be provably secure. On the other hand the HB-based protocols base their security on the hardness of the learning parity with noise (LPN) problem which is known to be NP-hard [4]. Although the HB-based authentication schemes are provably secure in strong settings, they all lack any notion of tamper-resilience. More recently, PUF-HB was proposed [1]. This scheme aimed at merging the two qualities of PUF and HB to produce an interesting hybrid. However, the work presented in [1] lacked any implementation results.

Our contribution: We present a proof of concept implementation for HB+PUF, a variant of PUF-HB which is also provably secure with a much simpler security reduction. The reduction only applies to active attacks which do not have a full man-in-the-middle control of the communication channel. HB+PUF resists the known man-in-the-middle attacks against the HB$^+$ scheme, and shows strong indications to resist a variety of these attacks. Due to the lightweight nature of PUFs and the HB$^+$ protocol the presented scheme proves suitable for lightweight applications. Our implementation takes advantage of the PUF circuit in order to produce the random bits typically needed for an HB-based authentication scheme. Note that the existence of a random number generator (RNG) is assumed in all HB-based protocols without taking into account the complexity of such a device. The overall circuit is shown to occupy less than 1000 gates.

The remainder of the paper is organized as follows. In Section 2 we review the PUF paradigm focusing on tristate PUF circuits. In Section 3 we give a review of the known HB-based authentication protocols. In Section 4 we define our notation and describe our proposed protocol. The security analysis is presented in Section 5. In Section 6 we describe our RNG construction and present analysis of its randomness. Section 7 describes the proof of concept implementation of the entire circuit. Finally, we present the conclusion in Section 8.

2 PUF

The idea behind a PUF is to have identical logical circuits observe different input-output behavior. This goal is achieved by taking advantage of the interchip variations which will vary from one circuit implementation to another. These variations are directly related to physical aspects of the fabrication environment. The reason one might seek to explore such a circuit is to prevent any ability to clone the system. Additionally, because of the high sensitivity of these interchip variations, it becomes difficult for an attacker to accurately reproduce the hardware. Another major advantage of the PUF's sensitivity is to prevent physical attacks on the system. Trying to tap into the circuit will cause a capacitance change therefore changing the output of the circuit. Similarly, trying to remove the outer layer of the chip will result in a change in the output of the circuit. We mention here that there are different realizations of PUFs. For example, surface PUFs were proposed in [36,44] and further developed in [40]. In this paper we focus our attention on the delay-based PUF first introduced in [14]. For the background information we closely follow the presentation of [1].

A delay-based PUF is a $\{0,1\}^n \rightarrow \{0,1\}$ mapping, that takes a n-bit challenge (a) and produces a single bit output (r). The basic idea of a PUF circuit is to create a race between two signals which originate from the same source. The original PUF circuit proposed in [14] utilizes multiplexers (MUXs) to implement the needed switches. In this work we use a different implementation of PUFs, in particular we use tristate PUFs proposed in [9]. Instead of having two interleaved delay paths, the tristate PUF uses two separate paths of delay units. This realization of a PUF has the advantage of requiring less gates and consuming less power. As shown in Figure 1 the delay units of a tristate PUF are constructed by respectively connecting the input and the output ports of two tristate buffers. The enable ports of these tristate buffers are connected to each other, with one of the buffers having an inverted enable input. This assures that only one of the two tristate buffers will be enabled for a particular input value. When a pulse is applied at the input of the delay unit, it will pass through only one of the two buffers. The enable input will determine which tristate buffer passes the pulse. This will change the amount of delay acquired by the signal according to the value of the enable input. To build the overall PUF circuit, the delay units are cascaded to construct a serial delay path. For a PUF circuit, we

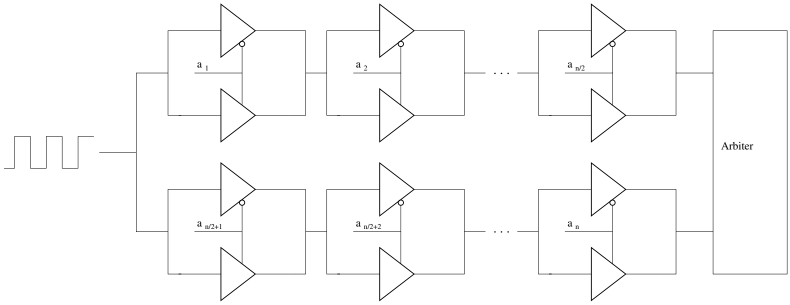


Fig. 1. PUF built with tristate buffers

need two separate delay paths. These delay paths are constructed as shown in Figure 1. The inputs of the two delay paths are generated by the same source while the outputs are fed to a flip-flop which we refer to as the *arbiter*. We assume the arbiter to be a positive edge-triggered flip-flop. The flip-flop has two inputs, the data and the clock. When the clock input has a rising edge the data input is captured at the output of the flip-flop. In the PUF setting, if the path that is connected to the data input has a smaller delay, then the output of the arbiter will be 1. Otherwise, the output will be 0.

In [9] the authors derive a linear delay model for the tristate PUF. The model is represented by an equation which gives the delay difference in terms of the challenge bits and the parameters of the PUF. Using their model we can relate the response bit r the challenge bits[1] a_i using the following function,

$$r = \mathrm{PUF}_Y(a) = \mathrm{sign}\left(\sum_{i=1}^{n}(-1)^{a_i} y_i + y_{n+1}\right).\text{[2]} \tag{1}$$

For simplicity we assume that n is even and we define $Y = [y_1, \ldots, y_{n+1}]$ where y_i denotes the imbalance in the signal propagation paths of the i^{th} stages and y_{n+1} denotes the imbalance in the delay between the upper and the lower paths and the setup time of the arbiter. These variables are dependent on the circuit itself and will vary from one PUF circuit to another. Sign$(x) = 1$ if $x \geq 0$, and 0 if $x < 0$. When the argument in sign(x) is zero the output becomes random. It is not possible to predict the behavior of the circuit when the two signals have the exact same mismatch. In fact, this behavior will happen to an even larger window of delay mismatch values. In such cases the PUF is said to be *metastable*. We will shortly address this issue by introducing the noise parameter ϵ_P. It is important to note here that the delay variations y_i will depend on the fabrication process of the PUF circuit. Therefore, one would expect these variables to follow a normal distribution. In particular, the y_i values will follow a Gaussian distribution of

[1] In this paper we use superscripts with parenthesis to refer to sequences of strings or bits, i.e. $a^{(j)}$. We use the subscripts to denote bits of a string, i.e. a_i.

[2] This model is almost identical to the model derived for MUX based PUFs.

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 could be represented using a linear inequality means that given a sufficient number of challenge-response pairs $(a^{(i)}, r^{(i)})$ for a single PUF, an attacker might be able to model the system using standard linear programming techniques [39,2]. To get an idea of these modeling attacks, for a 128-bit PUF with about 4000 challenge-response pairs one can model the PUF with an accuracy of less than 5%. Such an observation seems to completely undermine the idea of using a PUF. Although a PUF might be tamper-resilient, it still can be easily modelable. For theoretical and experimental results on modeling a PUF the reader is referred to [14,18,9]. In this paper we view this ability as an advantage that can help create a more practical system. The ability to model a PUF will eliminate the need to store a huge database to track all the deployed devices. The server can store the simple model captured by the real vector Y for each device. However, note that even with the best ability to model a PUF there will always be a level of inaccuracy caused by metastability. This observation is due to multiple reasons. First, the thermal noise will cause slight fluctuations in the internal variables therefore causing a change in the output. Second, the two signals propagating inside a PUF will sometimes enter a race condition such that the decision made by the arbiter will be random. Race conditions are in general the more dominant reason for metastability, and they will particularly happen when the delay difference between the two internal paths is less than the resolution of the arbiter. As a result, any PUF device can only be accurately modeled within a certain level. The best modeling schemes tested so far can provide an inaccuracy as low as 3% [29]. In our notation we denote the amount of error that exists in our best model of a PUF circuit with ϵ_P.

Next, we introduce an enhancement to the delay-based PUF. We use an n-bit non-zero binary string x to implement a linear bijection on the input challenge before it is fed into the PUF. Let a be the challenge string sent to the PUF. To produce the actual challenge string a' we treat x and a as elements of a finite field $GF(2^n)$ and compute the product $a' = xa \in GF(2^n)$. We next define a new PUF equation which takes this enhancement as well as the error in modeling the PUF into consideration[3]

$$\mathrm{PUF}_{Y,x,\epsilon_p}(a) = \mathrm{PUF}_Y(xa) \oplus \nu, \tag{2}$$

where $\nu = 1$ with probability ϵ_P and $\nu = 0$ with probability $1 - \epsilon_P$. The field multiplication may be implemented with low footprint using a simple linear feedback shift register (LFSR) based serial modular polynomial multiplier circuit. The choice of generating polynomial makes no difference in terms of the properties of the PUF device. Hence, for efficiency, low-weight polynomials (e.g. trinomials) may be used in the implementation [3].

[3] While this enhancement does not prevent modeling attacks, it will indeed have an affect on the man-in-the-middle attacks as we will see in Section 5.

3 LPN-Based Authentication Protocols

In this section we give a quick review of the LPN problem and the different HB authentication schemes which base their security on the hardness of LPN. We focus our attention on HB and HB$^+$. For a certain $\epsilon \in \left(0, \frac{1}{2}\right)$ the LPN problem is denoted by LPN$_\epsilon$, and stated as: *Given k random binary n-bit strings $a^{(j)}$ and the bits $z^{(j)} = a^{(j)} \cdot s \oplus \nu$ for some $s \in \{0,1\}^n$, 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 s.*[4]

The LPN problem is known to be NP-hard [4]. In [19], the authors show that the LPN problem is log-uniform and even hard to approximate within a ratio of 2. Kearns proved in [24] that the LPN problem is even hard in the statistical query model. The best known algorithm to solve the LPN problem is the BKW algorithm [5]. However, there has been a number of improvements on the algorithm with the best running time of $2^{O(n/\log n)}$ [12,30,33].

In the HB protocol [19], the tag and the reader share an n-bit secret string s. To authenticate the tag, the reader starts sending randomly generated n-bit challenge strings $a^{(j)}$. The tag responds with the bit $z^{(j)} = a^{(j)} \cdot s \oplus \nu$ 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 s after collecting n challenge-response pairs using Gaussian elimination. In [19], 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$_\epsilon$ 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 [20]. In HB$^+$ the tag and the reader share two n-bit strings s_1 and s_2. The tag starts the authentication session by sending a random n-bit string $b^{(j)}$. The reader then responds with $a^{(j)}$ just like the HB protocol. Finally the tag responds with $z^{(j)} = a^{(j)} \cdot s_1 \oplus b^{(j)} \cdot s_2 \oplus \nu$, where ν is defined as above. The protocol is proved 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 [20] and [23] 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 [16]. 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^{++} [7], HB-MP [34] and HB* [8]. A recent proposal is the HB$^\#$ [15]. In

[4] We follow the LPN formulation given in [23].

their work the authors propose a modified version of HB$^+$ which uses Toeplitz matrices rather than vectors for a shared secret. Under a strong conjecture the proposal 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 from the tag to the reader. Not that the GRS-MIM-model will essentially protect against the previously mentioned man-in-the-middle attack. Our work here is based on the more recent protocol PUF-HB [1] which introduces PUFs to the HB paradigm.

4 New Authentication Family: HB+PUF

In this section we present the proposed protocol. We will use $\mathcal{R}$ to denote the reader and $\mathcal{T}$ to denote the tag. n_1 and n_2 will be our security parameters. $\mathcal{T}$ and $\mathcal{R}$ are both characterized by the set of variables $(k, s_1, s_2, x, Y, \epsilon_P, \epsilon, u)$ where $s_1, x \in \{0,1\}^{n_1}$, $s_2 \in \{0,1\}^{n_2}$ and $Y = [y_1, y_2, \ldots, y_{n_1+1}]$ such that $y_i \in N(0,1)$ where $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. The noise parameters are $\epsilon, \epsilon_P \in \left(0, \frac{1}{2}\right)$. We use k to denote the number of rounds required for authentication. The last variable u is an integer in the range $[0, n]$ such that $\epsilon_f k \leq u$, where $\epsilon_f = \epsilon_P + \epsilon - 2\epsilon_p \epsilon$ denotes the total noise in the scheme.

With this notation we describe the basic authentication step. In every round, $\mathcal{T}$ randomly generates $b \in \{0,1\}^{n_2}$ and sends it to $\mathcal{R}$. Upon reception $\mathcal{R}$ replies with the challenge $a \in \{0,1\}^{n_1}$. Finally, $\mathcal{T}$ computes

$$z = a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,\epsilon_p}(a) \oplus \nu \ , \tag{3}$$

where $\nu = 1$ with probability ϵ and 0 with probability $1 - \epsilon$. Notice that this is very similar to the basic authentication step in HB$^+$. The only difference is that here we add a PUF operation. In order for $\mathcal{R}$ to authenticate $\mathcal{T}$, the same basic authentication step is repeated for k rounds. In every round $\mathcal{R}$ checks to see if $\mathcal{T}$'s response is equal to $(a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,0}(a))$. If the response is not equal to this term, $\mathcal{R}$ marks the response wrong. At the end of the k^{th} round, $\mathcal{R}$ authenticates $\mathcal{T}$ if and only if the number of wrong responses is less than 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 $\mathcal{R}_\sigma$ characterized by $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$. Following the notation in [23] we define $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle := 1$ iff $\mathcal{E}$ is authenticated by the reader, and is equal to 0 otherwise. The following protocol formalizes this interaction:

Protocol 1 (HB+PUF): $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle$

1. $\mathcal{R}_\sigma$ sets the counter $c = 0$
2. $\mathcal{E}$ sends $b \in \{0,1\}^{n_2}$ to $\mathcal{R}_\sigma$
3. $\mathcal{R}_\sigma$ choses $a \in \{0,1\}^{n_1}$ uniformly at random and sends it to $\mathcal{E}$
4. $\mathcal{E}$ sends z to $\mathcal{R}_\sigma$
5. if $z \neq a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,0}(a)$ then $c = c + 1$
6. Steps 2 through 5 are repeated for k iterations
7. If $c \leq u$ then $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle = 1$, otherwise it equals 0.

5 Security Analysis

In this section we show that the proposed protocol HB+PUF is at least as secure as the HB$^+$ protocol. We also discuss security against man-in-the-middle attacks. Finally, we consider the parameter selection to obtain a secure implementation. The reduction from HB+PUF to HB$^+$ is in fact very simple. As can be seen from Equation 3, the HB+PUF protocol utilizes all the terms of HB$^+$, and only adds a PUF operation. Therefore, it should be expected that the HB+PUF protocol can not be less secure than the HB$^+$ protocol. We now formalize this intuition by showing that any algorithm capable of successfully attacking the HB+PUF protocol can be used to successfully attack HB$^+$. The HB$^+$ protocol uses a tag $\mathcal{T}_\tau^+$ and a reader $\mathcal{R}_\tau^+$ both of which can be characterized by the string of variables $\tau = (k, s_1, s_2, \epsilon, u)$. The variables in τ are defined as we have done for the HB+PUF variables in Section 4. We also use $\langle \mathcal{E}, \mathcal{R}_\tau^+ \rangle$ to indicate an authentication session between any entity $\mathcal{E}$ and an HB$^+$ reader $\mathcal{R}_\tau^+$ using the HB$^+$ protocol. Similar to Protocol 1, $\langle \mathcal{E}, \mathcal{R}_\tau^+ \rangle = 1$ when the reader authenticates and 0 otherwise. This notation mostly follows the work presented in [23]. Recall from the previous section that in the HB+PUF protocol we use a tag $\mathcal{T}_\sigma$ and a reader $\mathcal{R}_\sigma$ both of which can be characterized by the string of variables $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$. We next state the reduction and keep the proof to Appendix 8. We prove the reduction in the active attacker model used to prove the security of the HB$^+$ protocol. In this model the attacker interacts with the tag in a learning session before he attempts to impersonate as the tag to an honest reader[5].

Theorem 1. *Let $\mathcal{A}$ be an algorithm which interacts with an honest HB+PUF tag $\mathcal{T}_\sigma$ for q authentication sessions to achieve* $\Pr[\langle \mathcal{A}, \mathcal{R}_\sigma \rangle = 1] > \delta$, *where $\mathcal{R}_\sigma$ is an honest HB+PUF reader. Then, there exists an algorithm $\mathcal{A}'$ which can interact with any HB$^+$ tag $\mathcal{T}_\tau^+$ for q authentication sessions to achieve* $\Pr[\langle \mathcal{A}', \mathcal{R}_\tau^+ \rangle = 1] > \delta$, *where $\mathcal{R}_\tau^+$ is an honest HB$^+$ reader.*

We point out that in the PUF-HB scheme the security reduction is much more involved since the secret s_1 is replaced by the PUF operation. Theorem 1 poses an immediate question of how the HB+PUF protocol behaves in relation to the the known man-in-the-middle attack against HB$^+$ [16]. Briefly, in this attack an adversary replaces all the challenges $\{a^{(j)}\}_{j=1}^k$ sent from the reader in a single authentication session by $\{a^{(j)} \oplus w\}_{j=1}^k$ where $w \in \{0,1\}^{n_1}$. The attacker knows that the challenges will interact with the secret s_1 through $a^{(j)} \cdot s_1$. At the end of the k rounds, if the reader authenticates the tag, then the adversary can deduce with very high probability that his changes did not affect the responses of the tag, and therefore $w \cdot s_1 = 0$. On the other hand, if the reader rejects the tag, then the adversary will know with a very high probability that $w \cdot s_1 = 1$. Repeating the same attack n_1 times will allow the adversary to collect n_1 linear equations

[5] We only need HB$^+$ to prove the reduction to the LPN problem. However, HB$^+$ is reduced to the LPN problem under the same attacker model used here.

containing s_1. Therefore, the adversary can use Gaussian elimination to solve for s_1 with a high probability.

As we pointed out earlier, one of the main reasons for such an attack to work is the linearity of the inner product operation. In our scheme the challenges $a^{(j)}$ are not only subjected to the inner product operation $a^{(j)} \cdot s_1$, but also to a PUF operation $a^{(j)} \cdot s_1 \oplus \mathrm{PUF}_{Y,x,\epsilon_p}(a^{(j)})$. With both operations being used, an adversary will need to find a way to modify the challenges such that he can deduce information about each of the two operations separately. To see why a PUF operation will help against the man-in-the-middle attacks, notice that on one hand the PUF is inherently non-linear due to the sign operation. Therefore, it will prevent against any simple man-in-the-middle attack trying to explore linearity, such as the attack in [16]. On the other hand, it has been shown in [1] that the probability distribution of two different challenges $a^{(1)}$ and $a^{(2)}$ yielding the same output from a PUF operation, will only depend on the Hamming distance between $a^{(1)}$ and $a^{(2)}$. This means that any successful man-in-the-middle attack would have to exploit the Hamming distances between different challenges. However, recall from the end of Section 4 that the PUF circuit used in HB+PUF implements a field multiplication over $GF(2^{n_1})$ with the secret string x. This multiplication will partially obfuscate the Hamming distance between different challenges. Therefore, the attacker's ability to deduce correlations between the inputs and the outputs of the PUF will be partially hindered.

Note that here we are talking with respect to the GRS-MIM model introduced in [15]. To protect against the most general class of man-in-the-middle attacks, we suggest adding a second PUF circuit to operate on the $b^{(j)}$ strings sent by the tag. In such a scheme the response of the tag would be

$$z = a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y_1,x_1,\epsilon_{p1}}(a) \oplus \mathrm{PUF}_{Y_2,x_2,\epsilon_{p2}}(b) \oplus \nu \ . \tag{4}$$

The suggested scheme will be more demanding in terms of hardware and power. However, we predict that it will be resilient against man-in-the-middle attacks.

We finish this section by discussing security parameters for an implementation of the design. As shown by Theorem 1 our protocol is at least as secure as the HB$^+$ protocol, which in turn is at least as hard as solving the LPN problem. All with respect to the active attacker model. In [30] the authors give a careful study of the BKW algorithm for solving the LPN problem. They conclude that the parameters first introduced for the HB$^+$ protocol by [20] and then by [23] do not provide sufficient security. In our implementation we follow the new parameters suggested by [30] and later adopted by [15]. To achieve 80-bits of security we choose $n_1 = 80, n_2 = 512$, $\epsilon_f = 0.15$ and $k = 200$. Note that ϵ_f is not a separate parameter but rather a result from ϵ_p and ϵ. In our implementation we will have $\epsilon_p = 0.15$ and $\epsilon = 0$.

6 PUF-Based RNG

In Section 2 we discussed the inherent metastability in a PUF circuit. As we pointed out earlier, these metastable states result from either environmental fluctuations, or race conditions which occur between the two propagating signals inside a PUF. In this section, we outline how metastability could be used to generate random bits. We note here that using a PUF circuit as an RNG is not a new idea. It has been previously proposed in [35]. In their design the authors use an LFSR to generate a stream of challenges. Each challenge is fed to the PUF multiple times in order to decide whether the challenge is metastable or not. Finally, the metastable outputs are used to extract randomness. In our approach, we take advantage of a PUF feedback setting. This approach will essentially remove any need to separately check each challenge for metastability. Therefore, decreasing the control logic, and increasing the throughput.

Our RNG design is based on a shift register feeding a PUF circuit in parallel. As we have concluded in Section 5 the size of the PUF and thus the size of the shift register will be 80 bits. The register is initialized to a random bit string. At every clock cycle the output of the PUF is fed back to the most significant bit of the shift register, while the least significant bit is discarded. This mode of operation will ensure a continuous stream of bits. Without metastability no randomness is expected to come out of this construction. Therefore, to assess the generated randomness we need to get a good estimate on the ratio of metastable points.

In order to get an estimate for the metastability ratio, we implemented the PUF circuit on a Xilinx XC2VP30 FPGA. In typical PUF implementations, extra precautions are taken to prevent metastability. However, we are interested in having a high level of metastability. This is the case, since we use the PUF in a *noisy* authentication scheme, and as an RNG. To help induce a higher level of metastability we allow close adjacency between the PUF circuit and other parts of the implementation. We carried out a restart test by collecting 1000 different bit streams. Each bit stream was collected after the system was reset and initialized to the same state. In a completely stable system, these bit streams would have to be identical. However, in a metastable system, every time a metastable point occurs these streams are expected to break into two groups, with each group following a different choice of the metastable point. After tracking all the bit streams we found that after 6400 bits all the 1000 streams were in completely different states, therefore suggesting the occurrence of 1000 metastable points. This yields an overall metastability ratio of about 15%. With this ratio, we can insure that the output always contains a metastable point by Xor-ing every 8 consecutive bits and using the result as the output of the RNG.

To verify the statistical quality of the RNG output, we collected a large number of bit streams and analyzed them with the NIST statistical test suite. As recommended by the NIST tools, every bit stream contained $20,000$ points. The test suite reports a *proportion* value, which reflects the ratio of bit streams which actually passed this particular test. The final results we obtained are shown in Table 1. The NIST tools return a statistical result where even a true random number generator could fail in some of the runs. We can conclude from the shown results that the proposed RNG is a reasonably good generator.

Table 1. NIST suite results

Test Name	Proportion
Frequency	100%
Frequency within block	100%
Longest run of ones in block	95%
Cumulative sum	100%
Runs	100%
Discrete Fourier Transform	100%
Non-overlapping template matching	95%
Overlapping template matching	97.5%
Maurer's Universal	100%
Approximate Entropy	97.5%
Serial	97.5%
Lempel-Ziv Complexity	100%

7 Implementation

The authentication scheme presented in Section 4 is implemented as shown in Figure 2. The PUF circuit is positioned at the center of the implementation to ensure tamper-resilience for the entire circuit. As we have verified from our FPGA implementations of a PUF circuit, any change in the surrounding hardware to the PUF circuit will result in changing the PUF's internal variables. We point out here that a PUF can easily protect against active side-channel attacks. However, for passive side-channel attacks a designer might have to resort to standard power balancing techniques [42,43,38]. Although effective, these technique will incur about 200 − 400% area overhead. A cost too high for lighweight implementations. Our authentication architecture runs in two different modes of operation during the entire protocol.

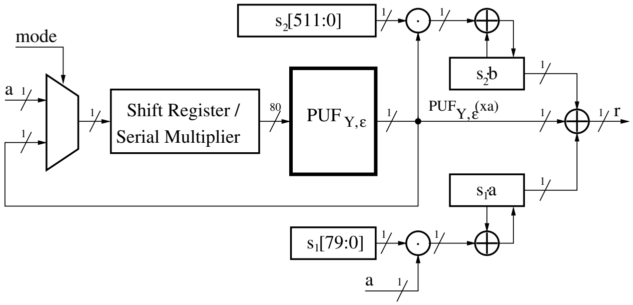


Fig. 2. PUF Authentication Scheme

RNG Mode: In this mode the PUF circuit acts as a random number generator. As explained in Section 6 the random string $b \in \{0,1\}^{512}$ is achieved using a shift register along with the PUF circuit. This shift register is initialized with the initialization value (IV) stored in an 80-bit ROM structure. The shift register will be serially initialized to (IV) in RNG mode. For the remainder of the RNG operation the serial input of the shift register will be fed back by the output of the PUF. Conveniently enough, we do not need to store the entire random string b generated by the PUF. As b is generated 1 bit at a time, we can serially compute the inner product $b \cdot s_2$, and at the same time serially transmit b to the reader. It is important to point out that in the RNG mode, the system will not be able to detect any active side-channel attacks. With a stream of random bits, the attacker's effect is gone undetectable. This will not be a major problem since any invasive attack on the circuit will permanently affect the PUF circuit. Therefore, as soon as the circuit is back to PUF mode, the attack can be detected. In the case where more gates are dedicated for security purposes, two separate PUF circuits can be used for authentication and random number generation.

PUF Mode: In this mode we perform the serial field multiplication xa which will be the input of the PUF. The hardware component *Shift Register/Serial Multiplier* shown in Figure 2 is used for this multiplication. The serial input of this shift register comes from the input a which is serially received. The field multiplication is realized through an LFSR serial multiplier, and is carried out in parallel with the inner product operation $a \cdot s_1$. These two operations will operate serially and will take about 80 cycles. The result of the field multiplication xa is fed to the PUF as the challenge input. The response bit of the PUF is then XOR-ed with the inner products $a \cdot s_1$ and $b \cdot s_2$. Finally, the response of the entire circuit r is transmitted to the reader. Note from the last section that the ratio of metastability was about 15%. This matches the overall desired noise. Therefore, there will be no need for an added noise parameter ϵ.

To estimate the gate count, HB+PUF was developed into Verilog modules and synthesized using the Synopsys Design Compiler. For the synthesis we used the TSMC 0.13 μm ASIC library. The circuit components and their gate count are explained as follows:

-**ROM Structure:** The IV for the RNG and the private keys s_1 and s_2 are stored inside the hardware and they are unique for each tag. Instead of utilizing flip-flops to store these values, we designed a ROM structure for low-area storage. To minimize the area usage, we used a design similar to a look-up table. Separate architectures are needed to store s_1, s_2 and IV. Since s_2 is 512 bits, s_1 and IV are 80 bits, we have 672 bits of total ROM area. Synthesis results show that 110 gates are required for this storage.
-**PUF Circuit:** In our authentication scheme, we utilize an 80-bit PUF circuit. As pointed out in Section 2 we use the tristate PUF implementation presented in [9]. This particular PUF implementation is of interest due to its low-power and low-area features. When we used the tristate PUF design our synthesis results showed the area of the PUF to be 350 gates. However, with custom circuit design, where each tristate buffer utilizes about a single gate, this number was reduced to 160 gates.
-**Shift register/Serial Multiplier:** The shift register has a total size of 80 bits. The structure also contains a number of XOR gates used to achieve the

field multiplication. In addition, 2-to-1 multiplexers are used to decide which inputs feed the individual flip-flops of the register. Synthesis results show that a total equivalent of 500 gates is needed for this structure.

-Serial inner products: The AND and XOR components shown in Figure 2 are utilized for serial inner product operations. The boxes labeled as $s_2 \cdot b$ and $s_1 \cdot a$ are single flip-flops storing the results of the inner products $s_2 \cdot b$ and $s_1 \cdot a$ of Protocol 1. They work as accumulators. In each clock cycle, one bit of s_2 and one bit of b pass through an AND gate and the result is XORed with the value in the accumulator flip-flop. The same procedure is repeated for s_1 and a. In the end, the results in the accumulator registers $s_2 \cdot b$ and $s_1 \cdot a$ are XOR-ed with the result of the $\mathrm{PUF}_{Y,\epsilon}(xa)$ and the result is sent to the output as r. The area for these operations is estimated at 50 gates.

-Control logic: The control logic for this scheme is quite simple. For the ROM structures storing s_1 and s_2, a single 9-bit counter is needed. Since the inner products for s_1 and s_2 are operated in parallel, a single counter is enough for both operations. For the RNG a 3-bit counter is needed to track the XOR-ing of each 8 consecutive bits. This can be interleaved with the inner product operation $s_2 \cdot b$. The architecture has only 2 modes of operation. Therefore, a single flip-flop would suffice to track the state of the hardware. The total area of the control block is estimated at about 150 gates.

The total area of the authentication hardware is 970 gates. This is below 1K gates, a typical threshold for the RFID's footprint allotted for security [37].

8 Conclusion

In this paper we presented a tamper-resilient and provably secure authentication scheme requiring less than 1K gates. We proved the security of our scheme against active attacks, and against known man-in-the-middle attacks. Moreover, the proposed scheme seems resilient against a more general class of man-in-the-middle attacks. We also demonstrated an efficient method for generating the random bits needed for our proposed protocol. This was done without incurring significant overhead to the hardware, and by reutilizing parts of the authentication circuit.

Our work here opens an interesting avenue for exploring cryptographic algorithms naturally supported by the hardware. For future work one might hope to explore modifications to the current protocol which might yield provable security against man-in-the-middle attacks. This problem has been addressed by multiple proposals, but no proof has been provided.

References

1. Hammouri, G., Sunar, B.: PUF-HB: A Tamper-Resilient HB based Authentication Protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 346–365. Springer, Heidelberg (2008)
2. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Mathematical Programming 71(2), 221–245 (1995)

3. Berlekamp, E.R.: Algebraic coding theory. McGraw-Hill, New York (1968)
4. 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)
5. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: Proceedings of STOC 2000, pp. 435–440. ACM, New York (2000)
6. Bogdanov, A., Leander, G., Knudsen, L.R., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT - An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
7. Bringer, J., Chabanne, H., Dottax, E.: HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In: Proceedings of SECPERU 2006, Washington, DC, USA, pp. 28–33. IEEE Computer Society, Los Alamitos (2006)
8. 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, pp. 23–26 (2007)
9. Ozturk, E., Hammouri, G., Sunar, B.: Physical Unclonable Function with Tristate Buffers. In: Proceedings of ISCAS 2008 (2008)
10. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L.: A Survey of Lightweight Cryptography Implementations. IEEE Design & Test of Computers – Special Issue on Secure ICs for Secure Embedded Computing 24(6), 522–533 (2007)
11. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong Authentication for RFID Systems Using the AES Algorithm. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156. Springer, Heidelberg (2004)
12. 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
13. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of CCS 2002, pp. 148–160. ACM, New York (2002)
14. 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)
15. Gilbert, H., Robshaw, M., Seurin, Y.: HB$^{\#}$: Increasing the Security and Efficiency of HB^{+}. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 361–378. Springer, Heidelberg (2008)
16. Gilbert, H., Robshaw, M., Sibert, H.: An Active Attack Against HB+ A Provably Secure Lightweight Authentication Protocol. IEE Electronic Letters 41, 1169–1170 (2005)
17. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., et al.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
18. Ozturk, E., Hammouri, G., Sunar, B.: Towards Robust Low Cost Authentication for Pervasive Devices. In: PERCOM 2008, Hong Kong, March 17-21 (2008)
19. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
20. 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)

21. Kaps, J., Gaubatz, G., Sunar, B.: Cryptography on a Speck of Dust. Computer 40(2), 38–44 (2007)
22. Kaps, J.-P., Sunar, B.: Energy Comparison of AES and SHA-1 for Ubiquitous Computing. In: Zhou, X., Sokolsky, O., Yan, L., Jung, E.-S., Shao, Z., Mu, Y., Lee, D.C., Kim, D.Y., Jeong, Y.-S., Xu, C.-Z. (eds.) EUC Workshops 2006. LNCS, vol. 4097, pp. 372–381. Springer, Heidelberg (2006)
23. 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)
24. Kearns, M.: Efficient Noise-Tolerant Learning from Statistical Queries. In: Proceedings of STOC 1993, pp. 392–401. ACM Press, New York (1993)
25. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
26. 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)
27. 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)
28. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, p. 196. Springer, Heidelberg (2007)
29. Lee, J.W., Daihyun, L., Gassend, B., Samd, 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)
30. Levieil, E., Fouque, P.: An Improved LPN Algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, p. 348. Springer, Heidelberg (2006)
31. Lim, C., Korkishko, T.: mCrypton-A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors. In: WISA, vol. 5, pp. 243–258
32. 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)
33. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subsetsum problem. In: APPROXRANDOM (2005)
34. Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. Comput. Networks 51(9), 2262–2267 (2007)
35. O'Donnell, C.W., Suh, G.E., Devadas, S.: Puf-based random number generation. Number 481 (November 2004)
36. Posch, R.: Protecting Devices by Active Coating. Journal of Universal Computer Science 4(7), 652–668 (1998)
37. Poschmann, A., Leander, G., Schramm, K., Paar, C.: New Ligh-Weight Crypto Algorithms for RFID. In: Proceedings of ISCAS 2007, pp. 1843–1846 (2007)
38. Regazzoni, F., Badel, S., Eisenbarth, T., Grobschadl, J., Poschmann, A., Toprak, Z., Macchetti, M., Pozzi, L., Paar, C., Leblebici, Y., et al.: 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)
39. Roos, C., Terlaky, T., Vial, J.-P.: Interior Point Methods for Linear Optimization, 2nd edn. Springer, Heidelberg (2005)

40. Skoric, B., Maubach, S., Kevenaar, T., Tuyls, P.: Information-theoretic Analysis of Coating PUFs. Cryptology ePrint Archive, Report 2006/101 (2006)
41. Standaert, F., Piret, G., Gershenfeld, N., Quisquater, J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Workshop on RFID and Lightweight Crypto, Graz, Austria (2005)
42. 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: Proceedings of ESSCIRC 2002, pp. 403–406 (2002)
43. Toprak, Z., Leblebici, Y.: Low-power current mode logic for improved DPA-resistance in embedded systems. In: ISCAS 2005, pp. 1059–1062 (2005)
44. Tuyls, P., Skoric, B.: Secret Key Generation from Classical Physics: Physical Uncloneable Functions. Philips Research Book Series. Springer, Heidelberg (2006)

Appendix A

Proof of Theorem 1

Proof. The basic operation of $\mathcal{A}'$ is to map a given $\tau = (k^+, s_1^+, s_2^+, \epsilon^+, u^+)$ characterizing the HB$^+$ tag and reader to $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$ used to characterize an HB+PUF tag and reader. Note that all the variables in the HB$^+$ protocol are still used in the same manner in the HB+PUF protocol. Therefore we can create $\sigma^+ = \left(k = k^+, s_1 = s_1^+, s_2 = s_2^+, x, Y, \epsilon = \epsilon^+, \epsilon_p = 0, u = u^+\right)$. The variable x is chosen randomly to be any string in $\{0,1\}^{n_1}$. The (n_1+1) real vector Y is chosen such that $y_i \in N(0,1)$. $\mathcal{A}'$ runs as follows: It initializes $\mathcal{A}$ and allows it to carry its communication with $\mathcal{T}_\tau^+$. In particular, $\mathcal{A}'$ passe the vector b sent by $\mathcal{T}_\tau^+$ to $\mathcal{A}$ which will reply with the vector a. Again $\mathcal{A}$ passes a back to $\mathcal{T}_\tau^+$. Finally, when $\mathcal{T}_\tau^+$ returns its response z, $\mathcal{A}'$ returns $\hat{z} = z \oplus \mathrm{PUF}_{Y,x,0}(a)$ to $\mathcal{A}$. The same step is followed for all q authentication rounds between $\mathcal{T}_\tau^+$ and $\mathcal{A}$. When $\mathcal{A}'$ wants to authenticate itself to $\mathcal{R}_\tau^+$, it again runs $\mathcal{A}$ in its authentication phase. $\mathcal{A}$ will start by sending the radnom string $b^{(i)}$. $\mathcal{A}'$ will pass the string directly to $\mathcal{R}_\tau^+$ which will respond with the vector $a^{(i)}$. $\mathcal{A}'$ passes $a^{(i)}$ back to $\mathcal{A}$. Finally, when $\mathcal{A}$ returns its response $z^{(i)}$, $\mathcal{A}'$ returns $\hat{z}^{(i)} = z^{(i)} \oplus \mathrm{PUF}_{Y,x,0}(a^{(i)})$ to $\mathcal{R}_\tau^+$. The algorithm $\mathcal{A}'$ repeats these steps for all k rounds of the authentication session, such that $i = 1 \dots k$.

To see why this will actually work, notice that in the first q rounds $\mathcal{A}$ is getting the response $\hat{z}$ which is effectively responses from a tag $\mathcal{T}_{\sigma^+}$. This means that at the end of the q authentication sessions $\mathcal{A}$ will have effectively communicated with $\mathcal{T}_{\sigma^+}$. In the authentication phase, when $\mathcal{A}'$ tries to authenticate itself to $\mathcal{R}_\tau^+$, it uses the algorithm $\mathcal{A}$ which will be trying to authenticate itself to $\mathcal{R}_{\sigma^+}$. Assuming that $\mathcal{A}$ will succeed in impersonating $\mathcal{T}_{\sigma^+}$ with probability larger than δ, then the responses returned by $\mathcal{A}$ which are $z^{(i)}$ will match the responses of an honest tag with probability larger than δ. However, this immediately implies that the responses returned by $\mathcal{A}'$ to $\mathcal{R}_\tau^+$ which are $\hat{z}^{(i)}$ and which differ from $z^{(i)}$ with the term $\mathrm{PUF}_{Y,x,0}(a^{(i)})$ will match the responses of an honest tag with probability larger than δ. Therefore, $\mathcal{A}'$ should also succeed in impersonating a tag $\mathcal{T}_\tau^+$ with probability larger than δ.

Threat Modelling in User Performed Authentication

Xun Dong, John A. Clark, and Jeremy L. Jacob

Department of Computer Science,
University of York
United Kingdom

Abstract. User authentication can be compromised both by subverting the system and by subverting the user; the threat modelling of the former is well studied, the latter less so. We propose a method to determine opportunities to subvert the user allowing vulnerabilities to be systematically identified. The method is applied to VeriSign's OpenID authentication mechanism.

1 Introduction

Criminals often seek to exploit a user's inability to distinguish the legitimate from the faked. *'Phishing'* attacks [1,13] are the most familiar examples; users are conned into taking actions that prove against their interests, typically resulting in the release of confidential and valuable information. Users may be regarded as complicit in such exploitation, but in many cases labelling of the user as 'the weakest link' merely covers up the fact that the systems are not designed to prevent such attacks or make them difficult. Users have reached their present exploitable state, aided and abetted by poor system design.

There is a deeper problem: there would appear to be little in the way of systematic *analysis* concerning the user's role in security. If users are now the weakest link then user-side threat modelling is as important as system-side threat modelling. In this paper we provide a user-focused threat identification approach for user authentication systems, particularly those used over the web. We hope our efforts will inspire further user-oriented threat modelling.

2 Background

Researchers have investigated security-relevant user behaviour addressing questions such as: How often are passwords reused [6]? How likely are users to choose secure passwords [6]? How effective are the security indicators shown on web browsers [3, 11, 14]? What factors influence users' trust in a website [2,4,5,7,8,9,10]? Why do users fall victim to phishing attacks and how can phishing interactions with users be modelled [3,4]? The practicality of such research may be limited. Findings may be too closely linked to current technology and

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 49–64, 2008.

could become invalid very quickly as systems, technologies, and user characteristics and behaviours evolve. Also, most studies are focused on specific problems, and there is little in the way of method to help system designers and security practitioners to systematically identify the threats to the users of systems.

Authentication is an interaction between a small group of entities (typically two) that aims to establish to each entity that the others have particular properties (often some notion of *identity*). We will focus on authentication between a *user* of a web service and an *external entity* (EE) that provides a service, or is a gateway to a service. We are interested in user-to-EE authentication and EE-to-user authentication. Both are typically achieved by proving ownership of certain objects (most typically secret knowledge such as passwords or encryption keys). While user-to-EE authentication is controlled by designers and consistently enforced by systems, designers often have little control over how EE-to-user authentication is carried out. Users must be able to first accurately establish the identity of the EE and then be able to judge whether this identity is entitled to request confidential information. The latter may seem straightforward for the user, but in some cases (as shown in section 4.2) it will be difficult. Failure of either part may lead to a user giving out authentication credentials to attackers.

Attacks against web authentication systems may be *passive* or *active*. Passive attacks do not require active victim involvement, often achieving their goal by analysing information available to attackers (e.g. that from public databases or websites, or even rubbish bin contents). Many are launched by insiders or people who have close relationships with the victims. Active attacks exploit the user's difficulty in authenticating EEs, requesting the user's authentication credentials whilst posing as trustworthy parties. Typical examples are phishing and pharming attacks. Mixed attacks are possible; some attacks have an initial passive phase to gather information and then use the information in a later active phase.

3 Vulnerabilities Exploited by Passive Attacks

3.1 Properties of Users' Authentication Credentials

A user and an EE share a collection of authentication credentials, typically including PINs, passwords, and so on. They also include an identifier unique to the user. Each credential can be classified along four axes: *mode*, *factor*, *assignment* and *losability*.

Mode. Primary or Emergency.

By *primary credentials* we mean those used to directly access assets and functionalities guarded or provided by external entities. Similarly *emergency credentials* denote those used to reset or recover the primary credentials. Attackers can masquerade as a user if they obtain primary or emergency credentials.

Factor something users know; something users possess; something users have access to; or characteristics of who users are.

The factor axis is based on the traditional way of classifying authentication approaches but with one addition. ‘Something users have access to’ is usually included in ‘something users possess’. We distinguish it because it has different implications for the security of an authentication system: it creates an authentication security dependency relationship. It is not directly possessed by a user, but the user holds the keys (authentication credentials) to access them. For example, an email account is not directly possessed by users, it is typically the property of an email provider, but a user with the correct password can access the content in that email account.

Assignment by the system; by the user; or by a third party;

Assignment by the system can ensure that certain security requirements are met (for example, that values of the authentication credentials are unique, and difficult to guess).

A user may not find the value assigned by system usable. User defined values may have high usability, but the system has limited control over whether security requirements are met.

When the value is assigned by a third party, the security properties depends on the behaviour of the third party. If the value of the authentication credential is predictable or easy to replicate, then this vulnerability could lead to the compromise of the current system.

Losability losable; or unlosable.

Losability indicates whether credentials are likely to be lost. If an authentication credential is losable, the authentication system must provide methods for users to recover their ability to authenticate.

3.2 Authentication Credentials Vulnerable to Passive Attacks

Passive attacks require authentication credentials to be exposed to third parties. Credentials known only to the system and the user have low exposure; credentials accessible to the general public (for example when a value has been published on a web page) have high exposure; otherwise the exposure is medium. The exposure level can be determined by considering the authentication credential factor basis, how its value is assigned, and the choice of authentication credentials by other authentication systems.

The exposure level of authentication credentials that are based on personal data and who users are, can only be medium or high. For example, a user’s date of birth or mother’s maiden name are known to close friends and relatives, and may even be available on a public database. Data that describes who users are, such as finger prints, are inevitably exposed to objects users have touched and all systems which use finger prints as authentication credentials.

Password exposure level can be low if it is the system that assigns the value, and the exposure level is uncertain prior to the assignment if it assigned by the user (since personal practices will differ).

Any authentication credential with mdium or high exposure level is vulnerable to a passive attack. Attackers may obtain data from those parties to whom the credentials have been exposed. For example there are companies that sell users' personal contacts and other personal information they collect. Hence it is very difficult to determine who has access to any high or medium exposure credential.

The complete set of a user's authentication credentials can be divided into subsets, each of which is sufficient to prove the identity of the associated identifier. The subsets always include at least one subset whose members are all primary authentication credentials, and may also contain other subsets of emergency authentication credentials. If those emergency credentials can be used to recover and/or reset the primary credentials and the primary credentials are all assigned by users, then the compromise of the emergency credentials is as serious as the compromise of primary credentials. To compromise one's account, attackers must obtain one of the subsets.

To determine whether attackers can obtain any of the subsets by applying passive attacks, analysts must find out the exposure level of each subset. The exposure level of a set of credentials is the minimum of its members. For example, if a subset has three members with exposure levels of high, medium, and low, then subset's exposure level is low. If a subset has high exposure level then the subset can be obtained using passive attacks; if a subset has a medium exposure level, then parties to which the credentials have been exposed can obtain the credentials using passive attacks. For any authentication system, designers should make sure there is no subset whose exposure level is high. When there are subsets whose exposure level is medium, then designers must assess how likely the parties to which the credentials are exposed are to launch attacks against the user. Design improvements can be taken if the exposure levels and security requirements warrant it.

3.3 The Authentication Security Dependency Graph

Authentication security dependency relationships. If compromise of system 'B' directly leads to the compromise of 'A' we say that the security of 'A' depends on the security of system 'B'. If any of the user authentication credentials are in the category 'what you have access to' or is created or assigned by a third party then effectively the designers may have created a dependency of the current system on the third party. For example, the access right to a secondary email account is often used as one of the emergency authentication credentials to reset or recover primary authentication credentials. In those cases, compromise of the email account allows attackers to gain access to the authentication system by resetting or retrieving the primary authentication credentials.

Drawing the dependency graph. Analysts should identify the dependency relationships and represent them in *an authentication security dependency graph.* Each node in the graph represents a system, and the start node of the graph is the system being designed. Directed edges are included from Node ‘A’ to Node ‘B’ exactly when Node ‘A’ depends on Node ‘B’. If ‘B’ also has such dependency relationships then the graph can be expanded further.

If the value of the authentication credentials which create such dependency relationships are assigned by users then the dependency relationships may be unpredictable and the graph cannot be determined. For example, if a user provides an email account as an emergency credential then it is impossible to predict all the email service providers that the current system will depend on.

Vulnerabilities. The dependency graph has two implications for the system being designed:

1. The security of the current system is equal to the security of the weakest system reachable in the graph; and
2. Obtaining authentication credentials to the weakest system propagates access back up the reachability chain.

The first implication means that the security of the current authentication system could be reduced if there is a weaker authentication system in the dependency graph. Many financial related websites have educated users to choose strong passwords and pay more attention to security indicators when accessing an authentication web page. Most users are likely to behave cautiously and securely when dealing with web sites they categorise as financial-related and important, but tend to use weak passwords and pay less attention to security for the rest [6]. However, the security of authentication of such financial-related web sites may not be strong, if some reachable node in the dependency graph from the financial-related site is treated less seriously. In fact, it is common for such web sites to ask users to provide a secondary email account as an emergency credential, while most users think that the security of their email account is less important than that of the financial-related web site.

The second implication means that the dependency relationships create new channels through which an authentication system may be attacked. The new channels are the ones that attackers could use to compromise the other systems in the graph. Moreover, the new channels can not be mitigated by the design of the authentication system.

Any dependency relationship should be viewed as a vulnerability, especially those which are unpredictable, and they should be avoided or minimised at the design stages. For unavoidable dependency relationships analysts should design the authentication system in a way that the authentication credentials that create the relationship are not used alone to prove identity. For example, access to the email account must be used together with a number of security questions to prove a user’s identity.

4 Vulnerabilities Exploited by Active Attacks

In the second stage, analysts should consider vulnerabilities that active attacks exploit. Our method considers phishing and pharming attacks.

4.1 Sensitivity of the Authentication Credentials

Section 4.1's reference to determining sensitivity level is OK in terms of indicating *what* needs to be done, but there is no information on *how* to do it, or indeed the practical feasibility of doing it. Sensitivity indicates the likelihood of the user being suspicious or alert when an external entity requests the authentication credential. If the user is very alert then the sensitivity is high, otherwise it is low. System designers should choose authentication credentials in a way that the sensitivity is as high as is practical.

User must be alert to the malicious request of authentication credentials. As mentioned in section 3.2, the user authentication credentials can be divided into several subsets. An analyst can predict the likely alertness of a user by examining the sensitivity level for each subset. The sensitivity of a subset is determined by the member with the highest sensitivity level. If there is a subset whose sensitivity is low, then there is a vulnerability. At least one subset should have its sensitivity at medium level, if possible every subset should have a high sensitivity level.

A credential data item's sensitivity level is subjective, and users' sensitivity levels for an data item may vary. In the process of determine the sensitivity level, analysts should use common sense, consulting a group of users if necessary.

4.2 Identify Potential Impersonating Targets

In active attacks, attackers need to impersonate a legitimate external entity (EE). It would be wrong to think that the impersonating target is only the current system. Attackers may impersonate three types of EE: the EE that the user has shared authentication credentials with; EEs that are entitled to request users' authentication credentials or initiate user-to-EE authentication; and the EEs that exist in the authentication dependency graph.

The first type are normally EEs with which the user has set up an account. However, there are some exceptions, for example, the single sign-on system such as OpenID. Here users set up an account with both an OpenID provider and the service provider website. The user shares its authentication credentials with the OpenID provider but not the service provider website.

The second type of EE can be difficult to identify. A company may have a number of websites, and users can use the same authentication credentials to access the services provided by all of them. It also happens when companies or organisations use the single sign-on system such as OpenID, in which users can use the same set of authentication credentials to access services provided by all participating companies. Another typical example is the credit/debit card authentication system: the card details are assigned and shared between the bank and the user, but online retailer websites may be entitled to request card details from its users. In all these examples there is no convenient mechanism for analysts and users to find out who are legitimate entities.

The third type of EEs can be identified by constructing the dependency graph.

Among the impersonating targets identified, if there are EEs whose authentication system designs cannot be influenced by the system designers, then there is a vulnerability that may be exploited. If the authentication system of such an EE is not designed or implemented properly, attackers might choose to impersonate that EE instead of impersonating the EE that designers can influence.

If EEs other than those users have shared credentials with may request them, and there is no reliable method to conveniently prove an entity is entitled to do so, then a vulnerability will be created – attackers could acquire users' credentials by claiming to be one of those further entities.

4.3 Active Attack Entry Point Analysis

Analysts should document all impersonating targets, and then carry out active attack entry point analysis for the targets that are within the control of the current system's designers. This is achieved by first identifying the entry points, and then analysing vulnerabilities at each entry point.

The lifecycle of authentication credentials. Figure 1 shows the states in the general lifecycle of authentication credentials and the transitions between them. The optional state and transitions between states are represented by dashed lines. There are seven states authentication credentials could be in:

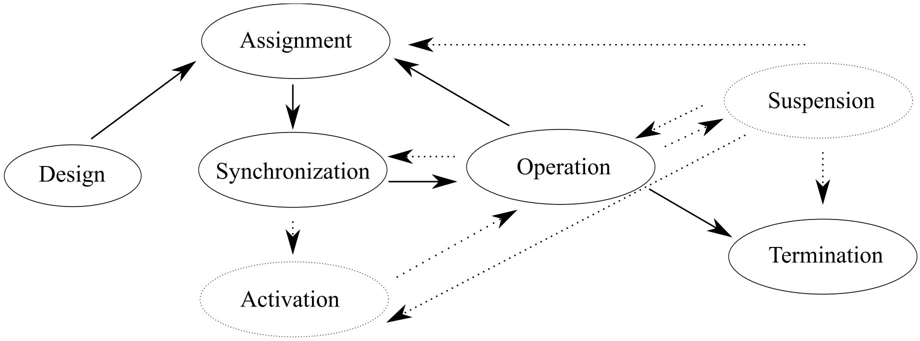


Fig. 1. The Lifecycle of Authentication Credentials

Design: Designers decide three things at this state : over which communication channel the authentication will be carried out; what authentication credentials should be used; and their lifecycle. The decision should be made based on the requirements for security, usability, constraints on economics and the properties of the authentication credentials (described in section 3.1). Our threat modelling should also be carried out in this stage.

Assignment: In this state the value(s) of the authentication credential(s) for a particular user will be created. Only the assigner knows the value of the credential(s).

Synchronisation: The party who assigned the value informs the other party of the value it has chosen. The time taken varies with the communication channel used. If users supply credential values via webpages then synchronisation could be immediate. For credentials exchanged by the postal system (e.g. a PIN number for a new cashpoint card), then the synchronisation could take a couple of days.

Activation: Some systems may require users to activate authentication credentials before they can be used.

Operation: Users supply their primary authentication credentials to authenticate themselves to external entities.

Suspension: The current authentication credentials temporarily cease to function, e.g. 'lockout' after three failed authentication attempts. Upon satisfying certain requirements authentication credentials can be tranformed to other states.

Termination: Here current authentication credentials permanently cease to function. The account may have been terminated by the system or the user.

Analysts should identify over which communication channels the authentication credentials are exchanged during each state and each transition between states. For the transitions originating from the operation state, analysts should also check whether the transition requires proof of identity. A vulnerability is created if the transition can be carried out without authentication, as attackers can request the transition without possessing any authentication credentials.

The activation and suspension states are optional. Any authentication credential should pass through the remaining five. However, transitions between states vary for different authentication credentials. A typical authentication credential's lifecycle starts at the design state before moving to assignment and synchronisation. Depending on the actual authentication system, there might be an activation state before the operation state. From operation it can reach four states: suspension, termination, assignment, and synchronisation. It often reaches assignment because the user or system decides to reset the current value of the authentication credentials. Not every system allows authentication credentials to transition from operation to synchronisation. But when it does, it is often due to loss of authentication credentials. For example, when a user forgets his password, the user ask the system to send him/her the password.

Transitions from the operation state can be triggered by events from both users and systems. When it is triggered by users, users normally are required to prove their identities by using emergency authentication credentials. On the other hand when the event is triggered by the system, the system will need to inform its users about the transition between states and may require users to complete the transition. If there is no rigorous EE authentication mechanism for users to reliably prove the EE's identity in the communication, then phishers could impersonate the trusted EE. That's why it is necessary to analyse the vulnerabilities of EE authentication within the communication between users and external entities.

Entry points. Active attacks can only obtain user's authentication credentials when they are exchanged. By using the lifecycle analysts can identify in which states and in which transitions this occurs and focus vulnerability analysis on those entry points. Using the lifecycle, we have identified the following six situations where a user's authentication credentials could be exchanged: 1)Synchronisation State; 2) Operation State; 3)state transition from operation to assignment; 4)state transition from operation to synchronisation; 5)state transition from suspension to assignment; 6)state transition from suspension to operation.

It is quite obvious the why a user's credentials are exchanged in both synchronisation and operation states. The transitions from the operation state can take place only when users have proved their identities. As a result users' emergency credentials will be exchanged. For example, when a user loses his primary credentials, such as password or USB token, the user needs to prove his identity to reset a new one.

Communication channels. The communication channel (CC) is a system or a method through which external entities can interact with human users. The most typical CCs are: 1) Emails; 2) Mobile phone messages; 3) Phone calls; 4) face to face communication; 5) webpages; 6) Instant Messenger; 7) Physical letters or notes. Authentication is a special type of interaction and it also operates via a CC. Each channel carries a different type of information, identifies entities in different ways, incvolves different agents, etc. As a result, EE-to-user authentication has different characteristics and vulnerabilities in different CCs. For a channel the following factors should be considered:

- How external entities are identified on the CC;
- How identifiers can be proved on the CC;
- Which Agents are involved in this CC.

The characteristics and vulnerabilities in EE-to-user authentication mainly depend on the communication channels over which the authentication is carried out. A limited number of CCs exist, so analysis of CCs could be carried out independently and the results can be used as a reference. Web page and email interactions are examined below:

Web page Interactions. Web pages are identified by their URLs. The integrity of website domain names are often proved by using an SSL/TSL certificate or an Extended Validation certificate. The agents involved in this communication channel can be classified as: client agents, server agents, and infrastructure agents. The client agents include:

- web browser
- operating platform (including the client computer's hardware and operating system)
- client side networking components: local router, gateway)

The server agents are the website servers. The infrastructure agents include:

- Domain Name Servers;
- Data delivery components;
- Certificate validation servers;

Email. An email is identified by its sender's address. To prove the email originates with the claimed sender SSL/TSL certificate can be used. The agents involved in this communication channel can be classified into three categories: Sender's agents, Receiver's agents, and infrastructure agents. The receiver's agents include:

- Email Client;
- Operating platform (including the client computer's hardware and operating system)
- Client's side networking components; (local router, gateway)
- IMAP/POP3 servers;

The sender's agents are responsible for delivery or relaying of the emails, for example, SMTP or MX Servers; The infrastructure agents include:

- Domain Name server;
- Data delivery components;
- Certificate validation server;

Entry points vulnerability analysis. For each attack entry point analysts determine the EE authentication vulnerabilities. These may arise due to:

- no reliable and sufficient authentication information is provided to users;
- users lack knowledge; and
- security design assumptions concerning users do not hold in practice;

Reliability and Sufficiency of Authentication Information. For successful EE-to-user authentication users must have reliable and sufficient authentication credentials. Because users mainly rely on authentication information presented to them to establish an external entity's identity, without them they can not accurately distinguish legitimate entities from the rest [4].

First analysts must determine the reliability of the authentication information provided. The authentication credentials the entity used to establish its identity, and all the agents involved on the channel on which the authentication system is implemented should be identified. This can be easily done by referring to the analysis of communication channels. The most important step is to check whether the compromise of any agents would make any EE authentication credentials untrustable. Then based on the protection of those agents, analysts can estimate how likely those agents would be compromised. If all agents were protected properly, then the reliability of that authentication credential would be high, otherwise it would be low.

Analysts can find out whether enough authentication information has given to users by checking first if users have been given the external entity's identifier. If the identifier's reliability is low given the previous analysis (which means it can be easily spoofed), then check whether users have been given additional reliable authentication credentials. If not, then users have insufficient information to carry out EE authentication.

Knowledge. Users need both technical and contextual knowledge to decide whether to release the credentials requested by an external entity. Technical knowledge helps users recognise and prove an external entity's identity based on given authentication information, while contextual knowledge help users to decide whether the external entity is entitled to request user's authentication credentials. Previous literature (e.g. [3]) have addressed only the user's need for technical knowledge, but contextual knowledge is equally important [4].

The technical knowledge required depends on the authentication communication channel. Users need knowledge to recognise external entity's identifier and understand associated authentication credentials. The knowledge must suffice to avoid falling victim to sophisticated spoofing techniques. The set of entities that are entitled to request the authentication credentials (AC) is obvious when the entity that requested the AC is the entity that users have shared ACs with. However, it would be ambiguous if the legitimate entity has delegated or shared the right to request the AC to other entities. Users need knowledge about how to distinguish (more than recognise) the set of entities that have been delegated from others. A typical example is an online shopping payment system. There is no clearly defined set of entities that could accept credit card details: online mechants, or some shops' own processing systems can all request card details. To steal users' financial credentials phishers could simply appear as a trustworthy online shop with its own payment processing system.

When an entity has a large number of identifiers users must know how to determine whether the identifier he/she currently sees is legitimate. Authentication on the telephone is an typical example of this. If a company has many telephone numbers users will have difficulty recognising whether the caller's number is one of those.

During the threat modelling practice, analysts should document the all expected knowledge from users. As those expected knowledge may change in the future especially those contextual knowledge. When those changes do happen, analyst can quickly identify the possible vulnerabilities by identifying the emerging knowledge gap.

Assumptions. The security of EE-to-user authentication assumes that users perform certain required actions correctly and consistently. System designers need to know how plausible such assumptions are. Results of existing empirical studies may prove useful [3,5,8,11,14,12,9] or further user studies may be carried out. If asssumptions prove implausible the system design must be altered. Users' behaviours are are affected by how systems are designed, education users receive, etc. As a result the legitimacy of assumptions on users is not static. Threat modelling analysts should document the user assumptions designers make for each entry point and periodically revalidate them. Invalid assumptions are clearly vulnerabilities.

4.4 External Entity Authentication in Communication Matters

Many active attacks lure victims by first impersonating the EEs in communication, such as masquerading as legitimate entities to send emails to users. The trust, expectation and perception constructed in communications could reduce users' ability to authenticate the EE in the following authentication session [4]. As a result, it is important to study vulnerabilities within the communication between legitimate entities and users.

The method used to analyse the vulnerabilities at entry points can be applied to analyse the vulnerabilities in communication. Here, there is more contextual knowledge users need to be aware of: 1) What are the communication channels the external entity would choose to initiate communication with users, if any? 2) In which circumstances the external entity would initiate communication with its users?

5 Case Study

We illustrate elements of the approach with reference to OpenID as a case study. OpenID is a decentralised, free and shared identity service, which allows Internet users to log on to many different web sites using a single digital identity, eliminating the need for a different user name and password for each site. It is increasingly gaining adoption among large sites, with organisations like AOL, BBC, Google, IBM, Microsoft, Orange, VeriSign, Yandex and Yahoo! acting as providers. We apply our method to analyse the default OpenID solution provided by VeriSign.

5.1 Passive Attack Analysis

Properties of authentication credentials. The complete set of user authentication credentials in this system is : {user name, password, access to a secondary email account} The property of the users' authentication credentials are listed in Table 1. Both authentication credentials are losable, but as long as not both credentials have been lost, their values can be recovered or reset.

Table 1. User Authentication Credential Properties

Authentication Credential	Mode	Factor	Assigned By	Losable
Password	Primary	Users know	User	True
Access to a chosen email account	Emergency	Users possess	User	True

Authentication credentials vulnerable to passive attacks. The password exposure level is uncertain, because it is user-assigned and there is no mechanism to ensure strong password choices. The exposure level for the access right to the email account is medium, because apart from the users and the system, the email service providers can also access the email messages in the email account.

Two sets of authentication credentials can be used to prove a user's identity, and each set has only one member: {password}, and {access to a chosen email account}. The exposure level for the password set is uncertain. This introduces a vulnerability, because the system cannot influence whether users choose weak passwords or reuse their passwords. Insiders are likely to be able to guess the password if it has been chosen poorly. The exposure for the second set medium, as its only member has medium exposure level. The parties to which the credential is exposed are limited, and it would be safe from general passive attacks.

Authentication security dependency graph. The authentication credential – access to a chosen email account – is in the category of what users have access to, so it has created the dependency relationships between VeriSign's OpenID authentication solution and the email providers which users have chosen. This relationship is unpredictable from the analysts' point of view, because there is no way to predict which email providers users would choose. Even worse, the access to the email account alone can complete the recovery and reset of the password.

According to our method, this design has at least two vulnerabilities: have created uncertain dependency relationships; and the system does not try to minimise the relationship by asking users to provide more authentication credentials together with the access to the chosen email account.

5.2 Active Attacks Analysis

Sensitivity of the authentication credentials. Both the password and the access right to the email account have high level of sensitivity.

Impersonating targets. The user has shared its authentication credential with VeriSign. The entities that are entitled to initiate the user authentication are not well defined and there is no mechanism for users to effortlessly and accurately know whether the entity that requests the use of OpenID authentication is legitimate or malicious. The email providers that existed in the dependency graph could also be the targets of impersonating. This is a vulnerability, as designers of VeriSign cannot patch or eliminate the vulnerabilities that existed in the EE-to-User authentication in the email systems.

Lifecycle of authentication credentials & entry points. Among all the possible targets, the only one designers can influence is VeriSign's authentication system. The lifecycle for the user's authentication credentials (including the communication channels) are shown in figure 2. There are three possible phishing attack entry points: 1) Synchronisation State; 2) Operation State; 3) State transition from operation to assignment;

5.3 Vulnerabilities at Each Entry Point

The methods and processes at each entry point analysis are the same, so for demonstration purposes, we explain only how the analysis is carried out for the operation state.

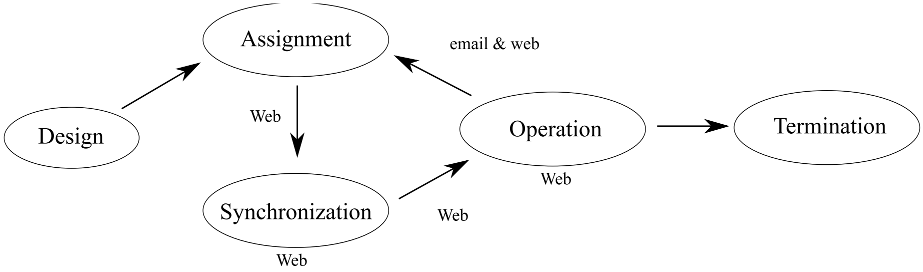


Fig. 2. The Lifecycle of Authentication Credentials

Operation state. In this state, the user first visits the service provider website and requests to sign in with his/her OpenID user ID. Assuming the user ID belongs to VeriSign, the user is directed to VeriSign's website, and is asked to enter the correct user name and password.

Reliability and Sufficiency of External Entity Authentication Credentials. In the operation state, all authentication actions are carried through the web page communication channel. VeriSign identifies itself on web by its domain name and URL. Its URL is `https://pip.verisignlabs.com/`. The compromise of the client side agents (operating system, web browser, networking components) and infrastructure agents could make the domain names displayed on the web browser no longer trustable. Among those agents, the client side agents are most vulnerable, least protected and exposed to the Internet. So its compromise is very likely, and the domain name alone is not reliable enough to identify the entity. VeriSign has used an SSL/TSL certificate to prove that the domain name is genuine. As a result, we can consider that VeriSign has provided sufficient and reliable authentication credentials.

Knowledge. To decide whether a URL belongs to VeriSign or any other entities (especially when some URLs are made to look as if they come from VeriSign) users must understand the syntax of URLs. Users also need knowledge to understand the SSL/TSL certificate to identify which entity really owns this domain. Both sets of knowledge are not possessed by many users [3]. This could be considered as potential vulnerability.

As for the contextual knowledge, it is not clear which websites are entitled to request users to use OpenID authentication. As a result, attackers could set up a phishing website which looks identical to VeriSign's and then lead users to this phishing website to steal the authentication credentials.

assumption. It is assumed that users check the URL and SSL certificate when they are in this state. Based on previous studies, we know a lot of users pay no attention to them. Users are unlikely to pay attention to the content of the SSL certificate, and they care only about their existence. [3,8,9,11,14]. As a result, the assumption on users has weak validity and vulnerability has been created.

5.4 Vulnerabilities in Communication

VeriSign uses only emails to communicate with its users. We will apply the same method that we used to analyse the entry point to analyse the vulnerability within the external entity authentication in the email system VeriSign uses.

Reliability and sufficiency of external entity authentication credentials. An email is identified by its sender's address. The compromise of client agents and any server agents could make the sender's address untrustable. In fact, the sender's address can be spoofed even without the compromise of any agent, as shown in [8]. So it is extremely unreliable to identify the real sender of the email. VeriSign has not used any other authentication credentials to prove that the email indeed comes from VeriSign, which has created a serious vulnerability and allows attackers to impersonate VeriSign in email communication channels. In conclusion, users have not been given reliable and sufficient authentication information to prove verisign originated the email.

Knowledge. Most users will be able to recognise the sender's email address. Users must have (contextual) knowledge needed of: the email address used by VeriSign to communicate with its users; and under which circumstances VeriSign would contact its users. VeriSign has not made clear to its users which email address it will use to communicate with users. As a result, email addresses whose semantic meaning is close to the email address VeriSign really uses could be accepted by users.

The last one is unclear as well, because VeriSign has not made this explicit to its users. As a result, it gives chances for phishers to create a scenario to lure victims to phishing websites.

Assumptions. It is assumed that users will check the email sender's address. This assumption is realistic and it is helped by the user interface design that users automatically read the sender and titles first.

6 Conclusions

User–side threat modelling is as important as system–side threat modelling, but it is much less well studied. This paper describes a method to systematically identity threats to web user authentication from user and social perspectives. Besides the VeriSign OpenID solution we have also used this method to identify threats to other user authentication systems: the UK national grid system, and Google websites. However, our method should not be viewed as complete; it is our initial effort towards developing a threat modelling method that can be used by system designers with moderate security knowledge. In future we will further refine this method and evaluate its usability by system designers. The provision of analysis tools for investigating threats to the user is important and we recommend the area to the research community.

References

1. Anti-phishing work group home page (2007), http://www.antiphishing.org/
2. Flinn, S., Lumsden, J.: User perceptions of privacy and security on the web. In: The Third Annual Conference on Privacy, Security and Trust (PST 2005), St. Andrews, New Brunswick, Canada, October 12-14 (2005)
3. Dhamija, R., Tygar, D., Hearst, M.: Why phishing works. In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, ACM Special Interest Group on Computer-Human Interaction, pp. 581–590 (2006)
4. Dong, X., Clark, J.A., Jacob, J.: A user-phishing interaction model. In: Conference on Human System Interaction (2008)
5. Downs, J.S., Holbrook, M.B., Cranor, L.F.: Decision strategies and susceptibility to phishing. In: SOUPS 2006: Proceedings of the second symposium on Usable privacy and security, pp. 79–90. ACM Press, New York (2006)
6. Florencio, D., Herley, C.: A large-scale study of web password habits. In: WWW 2007: Proceedings of the 16th international conference on World Wide Web, pp. 657–666. ACM Press, New York (2007)
7. Friedman, B., Hurley, W.D., Howe, D.C., Nissenbaum, H., Felten, E.W.: Users' conceptions of risks and harms on the web: a comparative study. In: CHI Extended Abstracts, pp. 614–615 (2002)
8. Jagatic, T., Johnson, N., Jakobsson, M., Menczer, F.: Social phishing. ACM Communication (October 2007)
9. Jakobsson, M., Tsow, A., Shah, A., Blevis, E., Lim, Y.-K.: What instills trust? a qualitative study of phishing. In: USEC 2007 (2007) (Extended abstract)
10. Nikander, P., Karvonen, K.: Users and trust in cyberspace, pp. 24–35 (2001)
11. Schechter, S., Dhamija, R., Ozment, A., Fischer, I.: The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies. In: 2007 IEEE Symposium on Security and Privacy (2007)
12. Whalen, T., Inkpen, K.M.: Gathering evidence: use of visual security cues in web browsers. In: GI 2005: Proceedings of Graphics Interface 2005, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, pp. 137–144. Canadian Human-Computer Communications Society (2005)
13. Wikipedia. Phishing. web, http://en.wikipedia.org/wiki/Phishing
14. Wu, M., Miller, R.C., Garfinkel, S.L.: Do security toolbars actually prevent phishing attacks? In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 601–610. ACM Press, New York (2006)

Access with Fast Batch Verifiable Anonymous Credentials

Ke Zeng

NEC Laboratories, China
zengke@research.nec.com.cn

Abstract. An anonymous credential-based access control system allows the user to prove possession of credentials to a resource guard that enforce access policies on one or more resources, whereby interactions involving the same user are unlinkable by the resource guard. This paper proposes three fast batch verifiable anonymous credential schemes. With all three schemes, the user can arbitrarily choose a portion of his access rights to prove possession of credentials while the number of expensive cryptographic computations spent is independent of the number of accessx rights being chosen. Moreover, the third anonymous credential scheme is not only fast batch verifiable but also fast fine-grained revocable, which means that to verify whether an arbitrarily chosen subset of credentials is revoked entails constant computation cost.

Keywords: anonymous credential, batch verification, fine-grained revocation, pairing.

1 Introduction

The protection of consumer privacy in access-control-based applications is a challenge that can be postponed no longer [1]. Access control simply means the act of determining if a particular right, such as access to some resource, can be granted to the presenter of a particular credential [2]. The access control system that this paper describes has a pseudonym authority (PA), resource holder (RH), resource guard (RG), and user as types of players. The PA issues a pseudonym to the user. The RH manages resources and, by issuing credentials, grants resource access rights to the user. The RG enforces access policies on the resources of one or more RHs and, by verifying the pseudonym and the credentials of a user, admits or denies the user access to the resources according to the RG's access control policies.

It's interesting to note the common practice that the applications try to collect as much personal information as possible from users, due to the incentive to price discriminate [3]. Hence, the user will frequently confront challenges in deciding how many resources are accessible and how much privacy is compromised. The user's decision may vary from service to service, from time to time, and from person to person. It's thus important to devise efficient anonymous credential schemes that enable verifying an arbitrarily chosen set of user credentials such that only absolutely necessary access rights (that are sensitive user information) are exposed to the access-control-based applications, with as few as possible computational cost.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 65–80, 2008.

Camenisch et al. [4, p. 2] have proposed seven desirable properties of an anonymous credential system, i.e., existential unforgeability, unlinkability, traceability, separability for resource holders[1], revocability, multi-show/one-show credentials, non-transferability. In addition to these, this paper highlights four desirable properties: batch verifiable, fine-grained revocable, and their *fast* versions.

(i) Batch verifiable. A user can arbitrarily select a portion of his access rights and prove possession of credentials to the RG without exposing other access rights.

(ii) *Fast* batch verifiable. In addition to (i), the number of expensive cryptographic computations spent and the pieces of data generated by proving possession of credentials are independent of the number of access rights being chosen. Scalar multiplication, modular exponentiation, and pairing evaluation are in general considered expensive.

(iii) Fine-grained revocable. Any one credential of the user can be revoked while leaving his other credentials untouched. In other words, revocation of credentials is on a per-user per-access-right basis.

(iv) *Fast* fine-grained revocable. The RG may need to ascertain whether a subset of the user's credentials has been revoked. In such case, in addition to (iii), the number of expensive cryptographic computations spent and the pieces of data generated by proving that the subset of credentials are not revoked are independent of the size of the subset being chosen.

Our Contribution

To the best of our knowledge, this work is the first that addresses an anonymous credential system achieving the *fast* batch verifiable property and the *fast* fine-grained revocable property. Three pairing-based anonymous credential schemes are presented. The first scheme achieves the *fast* batch verifiable property in the random oracle model. The second and third schemes achieve the *fas*t batch verifiable property in the standard model. In particular, the third scheme achieves the *fast* fine-grained revocable property as well.

2 Related Work

The anonymous credential system has been extensively studied in academia, resulting in many schemes, including those of [1, 4 - 14], just to name a few.

Chaum et al. [5] first introduced the scenario with multiple users that request credentials from resource holders then anonymously present credentials to resource guards without involving the resource holders online. The schemes proposed in [6] and [7] are based on having a trusted third party involved in all interactions.

Persiano et al. [10,11] proposed two anonymous credential schemes. The work of [10] is based on a chameleon certificate. The work of [11] is based on Strong RSA assumption. However, these schemes rely on inefficient zero-knowledge proofs, such

[1] Here it should be noted that the term Organization is originally utilized in [4]. The Organization not only issues credentials but also verifies credentials. In this regard, we logically divide Organization into Resource Holder and Resource Guard.

as proving knowledge of a double discrete logarithm, which is too expensive to be adoptable in practice [15].

Camenisch et al. [4, 12] proposed two efficient anonymous credential schemes, wherein [4] is based on Strong RSA assumption and [12] is based on LRSW assumption.

Most recently, Akagi et al. 14 proposed a q-SDH assumption-based anonymous credential scheme. This scheme is more efficient than the above ones because it utilizes a simplified system model (we will elaborate this simplified system model in Section 5.1).

However, none of the previous work presents a fast batch verifiable anonymous credential scheme. As Camenisch et al. recently pointed out, it is as yet an open problem to find a fast batch verification scheme for anonymous credentials [16].

Moreover, none of the previous work explicitly addresses the fine-grained revocable property. And none presents a fast fine-grained revocable anonymous credential scheme.

3 Preliminaries

3.1 Notations and Number-Theoretic Preliminaries

If $\mathcal{S}$ is a finite set, $x \in_R \mathcal{S}$ denotes that x is chosen from $\mathcal{S}$ uniformly at random. Let $\Omega(\cdot)$ be an arbitrary Boolean predicate, i.e., a function that, upon input of some string ς, outputs either TRUE or FALSE. By $\varsigma \leftarrow A(x) : \Omega(\varsigma)$ we denote that $\Omega(\varsigma)$ is TRUE after ς was obtained by running algorithm $A(\cdot)$ on input x. A function $adv(k)$ is said to be negligible if for every positive polynomial $p(\cdot)$ and sufficiently large k, $adv(k) < 1 / p(k)$.

Throughout this paper, we use the traditional multiplicative group notation, instead of the additive notation often used in elliptic curve settings.

Let $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ be two finite cyclic groups with additional group $\mathcal{G} = \langle g \rangle$ such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathcal{G}| = p$ where p is a large prime. Let $\mathbb{G}_1^*$ denote $\mathbb{G}_1 \setminus \mathcal{O}$ where $\mathcal{O}$ is the identity of $\mathbb{G}_1$. Bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathcal{G}$ is a function, such that: Bilinear, for all $h_1 \in \mathbb{G}_1$, $h_2 \in \mathbb{G}_2$, and for all $a, b \in \mathbb{Z}_p$, $e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$; Non-degenerate, $\exists h_1 \in \mathbb{G}_1$, $\exists h_2 \in \mathbb{G}_2$ such that $e(h_1, h_2) \neq \mathcal{I}$ where $\mathcal{I}$ is the identity element of $\mathcal{G}$; and Computable: there exists an efficient algorithm for computing e.

We suppose there is a setup algorithm $Setup(\cdot)$ that, upon input of security parameter 1^k, outputs the above settings of the bilinear map and writes this as $(p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e) \leftarrow Setup(1^k)$.

q-SDH Assumption. For all probabilistic polynomial-time (p.p.t.) adversaries $\mathcal{A}$, $adv(k)$ defined as follows is a negligible function:

$$(p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e) \leftarrow Setup(1^k); a \in_R \mathbb{Z}_p;$$
$$\Pr\left[(x,y) \leftarrow \mathcal{A}(g_2{}^a, g_2{}^{a^2}, \cdots, g_2{}^{a^q}) : x \in \mathbb{Z}_p \wedge y = g_1{}^{1/(a+x)}\right] = adv(k)$$

The q-SDH assumption has been shown to hold in generic bilinear groups by Boneh et al. [17].

3.2 Honest-Verifier Zero-Knowledge (HVZK) Proof

Let $KP\left\{(\varsigma):\ \Omega(\varsigma)\right\}$ denote a zero-knowledge proof instance between a prover and a verifier, where their common input is a predicate $\Omega(\cdot)$, and the prover's secret input is a string ς. $KP\left\{(\varsigma):\ \Omega(\varsigma)\right\} = \text{TRUE}$ denotes the case that the verifier is convinced that $\Omega(\varsigma)\text{=TRUE}$, and $KP\left\{(\varsigma):\ \Omega(\varsigma)\right\} = \text{FALSE}$ denotes the case otherwise. The honest-verifier zero-knowledge (HVZK) proof has been extensively studied during the past two decades, resulting in many efficient techniques [19 - 21].

In particular, we elaborate $KP\left\{(t,\tau,z):\ e(\mathrm{T},A) = e(h^\tau \cdot \mathrm{T}^{-z}, g_2) \wedge \mathrm{T} = t^\tau\right\}$ as below Protocol 1, which is an HVZK proof of knowledge of $t \in \mathbb{G}_1$, $\tau \in \mathbb{Z}_p$, and $z \in \mathrm{Z} \subseteq \mathbb{Z}_p$ such that congruence $e(t^\tau, A \cdot g_2{}^z) = e(h^\tau, g_2)$ holds..

Protocol 1

Bilinear map $(p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e) \leftarrow Setup(1^k)$ and $A \in \mathbb{G}_2$ are system parameters. All these plus $h \in \mathbb{G}_1$ and $\mathrm{T} = t^\tau \in \mathbb{G}_1$ are common inputs to the prover and verifier.

The prover's secret input is $t \in \mathbb{G}_1$, $\tau \in \mathbb{Z}_p$, and $z \in \mathrm{Z} \subseteq \mathbb{Z}_p$.

The goal of Protocol 1 is to prove knowledge of t, τ, and z, such that $e(\mathrm{T},A) = e(h^\tau \cdot \mathrm{T}^{-z}, g_2)$ and $\mathrm{T} = t^\tau$, i.e., $e(t^\tau, A \cdot g_2{}^z) = e(h^\tau, g_2)$.

1) The prover selects $(\alpha,\beta) \in_R \mathbb{Z}_p{}^2$, computes $\mathrm{R} = e(h^\alpha \cdot \mathrm{T}^{-\beta}, g_2) \in \mathcal{G}$, and sends R to the verifier.
2) The verifier selects a challenge $c \in_R \mathbb{Z}_p$ and sends c to the prover.
3) The prover computes $(s_\tau = \alpha - c \cdot \tau, s_z = \beta - c \cdot z) \in \mathbb{Z}_p{}^2$, and sends (s_τ, s_z) to the verifier.

The verifier is convinced iff $\mathrm{T} \in \mathbb{G}_1^*$ and $\mathrm{R} = e(h^{s_\tau} \cdot \mathrm{T}^{-s_z}, g_2) \cdot e(\mathrm{T},A)^c$.

Also notice that Protocol 1 is a secure three-move identification scheme [22].

Claim 1. The number of expensive cryptographic computations spent and the pieces of data generated by Protocol 1 are constant.

4 Fast Batch Verifiable Anonymous Credential

Describing anonymous credential scheme as five algorithms: $PAKeyGen(\cdot)$, $RHKeyGen(\cdot)$, $UserEnroll(\cdot)$, $GCred(\cdot)$, and $VCred(\cdot)$, we formalize the notions of batch verification and fast batch verification.

$PAKeyGen(\cdot)$. This probabilistic algorithm takes as input the security parameter 1^k, and returns the PA public key pk^{PA} and private key sk^{PA}.

$RHKeyGen(\cdot)$. This probabilistic algorithm takes as input pk^{PA} and access right R_{ij} that is under control of RH_i, and returns access-right public key pk_{ij}^{RH} and private key sk_{ij}^{RH} of access right R_{ij}.

$UserEnroll(\cdot)$. This probabilistic algorithm takes as input pk^{PA}; sk^{PA}; $n_{\max}$, which is the maximum number of admissible users; and a user identity U_l; and returns user key x_l and user pseudonym nym_l.

$GCred(\cdot)$. The credential issuance algorithm takes as input pk^{PA}, nym_l, x_l, access right R_{ij}, corresponding access-right public key pk_{ij}^{RH} and private key sk_{ij}^{RH}, returns credential $Cred_{ijl}$.

$VCred(\cdot)$. The credential verification algorithm takes as input pk^{PA}, nym_l, x_l, a set of access rights $\left\{R_{ij}\right\}$, corresponding access-right public keys $\left\{pk_{ij}^{RH}\right\}$, and purported credentials $\left\{Cred_{ijl}\right\}$. It decides whether to accept or to reject the credentials, and returns TRUE or FALSE, respectively.

Definition (Batch Verifiable Anonymous Credentials). An HVZK knowledge proof instance $KP\left\{(nym_l, x_l, \left\{Cred_{ijl}\right\}) : VCred(pk^{PA}, nym_l, x_l, \left\{R_{ij}\right\}, \left\{pk_{ij}^{RH}\right\}, \left\{Cred_{ijl}\right\})\right\}$ is a batch verification of anonymous credentials if the following two conditions hold:

- If for all R_{ij}, $VCred(pk^{PA}, nym_l, x_l, R_{ij}, pk_{ij}^{RH}, Cred_{ijl}) = \mathrm{TRUE}$, then given the honest prover $KP\left\{\begin{matrix}(nym_l, x_l, \left\{Cred_{ijl}\right\}) : \\ VCred(pk^{PA}, nym_l, x_l, \left\{R_{ij}\right\}, \left\{pk_{ij}^{RH}\right\}, \left\{Cred_{ijl}\right\})\end{matrix}\right\} = \mathrm{TRUE}$
- If for any R_{ij}, $VCred(pk^{PA}, nym_l, x_l, R_{ij}, pk_{ij}^{RH}, Cred_{ijl}) = \mathrm{FALSE}$, then $adv(k)$ defined as follows is a negligible function:

$$\Pr\left[KP\left\{\begin{matrix}(nym_l, x_l, \left\{Cred_{ijl}\right\}) : \\ VCred(pk^{PA}, nym_l, x_l, \left\{R_{ij}\right\}, \left\{pk_{ij}^{RH}\right\}, \left\{Cred_{ijl}\right\})\end{matrix}\right\} = \mathrm{TRUE}\right] = adv(k)$$

With this definition, it's easy to see that most existing anonymous credential schemes can conduct batch verification of anonymous credentials, e.g. the scheme as per [9]. Specifically for some of the existing schemes, adopting batch verification method for modular exponentiations due to Bellare et al. [23] may yield alternative approaches, see for instance [24] which presented general approach to batch verification of short signatures. Whereas, neither approach is *fast*, as analyzed below, as per our definition for fast batch verification of anonymous credentials.

Definition (Fast Batch Verifiable Anonymous Credentials). The above batch verification of anonymous credentials is said to be fast if, for all $\tilde{\mathcal{R}} \subseteq \left\{R_{ij}\right\}$, the number of expensive cryptographic computations spent and the pieces of data generated by

$$KP\left\{(nym_l, x_l, \left\{Cred_{ijl}\right\}) : VCred(pk^{PA}, nym_l, x_l, \tilde{\mathcal{R}}, \left\{pk_{ij}^{RH}\right\}, \left\{Cred_{ijl}\right\})\right\} = \text{TRUE}$$

are independent of $\left|\tilde{\mathcal{R}}\right|$.

5 Fast Batch Verifiable Anonymous Credential Scheme

5.1 Scheme I

Our first fast batch verifiable anonymous credential scheme (Scheme I) is based on a simplified system model. This simplified system model has four types of players: the portal service (PS) that not only manages pseudonyms but also manages access rights on behalf of the RHs, the RH that is transparent to the user, the RG, and the user. It's notable that the work of [14] is based on the same simplified system model.

Notice that with this simplified system model, algorithm $RHKeyGen(\cdot)$ merges with algorithm $PAKeyGen(\cdot)$ as $PSKeyGen(\cdot)$, and algorithm $UserEnroll(\cdot)$ merges with algorithm $GCred(\cdot)$.

***PSKeyGen(·)*:** The PS calls $Setup(\cdot)$ according to the security parameter 1^k, chooses a full-domain hash function $Hash(\cdot) : \left\{0,1\right\}^* \to \mathbb{G}_1$ that is viewed as a random oracle by the security analysis; and chooses $a \in_R \mathbb{Z}_p$; and computes $A = g_2^{\ a} \in \mathbb{G}_2$.

The PS's public key is $pk^{PS} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e, A, Hash(\cdot))$ and private key is $sk^{PS} = a$.

***GCred(·)*:** In order to be granted access rights R_i by the PS, a user who has trustworthy identity U_a carries out the following access-rights-granting protocol with the PS:

2.a) The user sends its identity U_a to the PS and the PS queries its database for stored key z of U_a. Iff it does not find a match, the PS selects $z \in_R \mathbb{Z}_p$ for U_a and stores (U_a, z) in its database.

2.b) The PS computes $t_i = Hash(R_i)^{1/(a+z)}$, and sends user key z and credential t_i to the user.

2.c) User U_a verifies that $e(t_i, A \cdot g_2^{\ z}) = e(Hash(R_i), g_2)$.

Fast Batch Verification of Anonymous Credentials: Suppose the user U_a has been granted access rights $\left\{R_1, R_2, \ldots, R_\Upsilon\right\}$. Without loss of generality, suppose one subset $\left\{r_j\right\} \subseteq \left\{R_1, R_2, \ldots, R_\Upsilon\right\}$ of the user's access rights matches one of the RG's policies $Pol = \left(\left\{r_j\right\}, \mathrm{H} = \prod_j Hash(r_j), \ldots\right)$. Let t_j denote the user's credential that corresponds to access right r_j.

Define $VCred(\cdot)$ as $e(\prod_j t_j, A \cdot g_2^{\ z}) \overset{?}{=} e(\prod_j Hash(r_j), g_2)$. In order to prove to the RG that he meets policy Pol, the user carries out the following anonymous access control protocol with the RG:

3.a) The user U_a computes $\mathrm{t} = \prod_j t_j$, selects $\tau \in_R \mathbb{Z}_p$, and computes batch verifiable anonymous credential $\mathrm{T} = \mathrm{t}^\tau$.

3.b) User U_a notifies the RG of the matching of policy Pol and sends T to the RG.

3.c) RG interacts with U_a for $KP\left\{(\mathrm{t}, \tau, z):\ e(\mathrm{T}, A) = e(\mathrm{H}^\tau \cdot \mathrm{T}^{-z}, g_2) \wedge \mathrm{T} = \mathrm{t}^\tau\right\}$ utilizing Protocol 1.

3.d) If $KP\left\{(t, \tau, z):\ e(\mathrm{T}, A) = e(\mathrm{H}^\tau \cdot \mathrm{T}^{-z}, g_2) \wedge \mathrm{T} = \mathrm{t}^\tau\right\} = \mathrm{TRUE}$, the RG is convinced.

5.1.1 Scheme I Security

We formalize the existential unforgeability of Scheme I as an adaptive chosen-key and adaptive chosen-access-right game. In this model, the adversary $\mathcal{A}$ is given a single public key. His goal is the existential forgery of a batch verifiable anonymous credential. The adversary's advantage, $\mathrm{ADV}_{\mathcal{A}}^{SchemeI}$, is defined as his probability of success in the game.

Definition (Existential Unforgeability of Scheme I). An adversary $\mathcal{A}$ $(N, t, \mathrm{K}, \Gamma, \varepsilon)$-breaks the existential unforgeability of the N-user Scheme I in the adaptive chosen-key and adaptive chosen-access-right model if $\mathcal{A}$ runs in time at most t, makes at most K queries to the hash function, issues at most Γ credential queries to the challenger, and $\mathrm{ADV}_{\mathcal{A}}^{SchemeI}$ is at least ε.

Theorem 1.1. Proposed Scheme I is secure against existential forgery in the random oracle model under q-SDH assumption.

Corollary 1.1. Proposed Scheme I is a batch verification scheme for anonymous credentials.

Corollary 1.2. Based on Claim 1, proposed Scheme I is a *fast* batch verification scheme for anonymous credentials.

Now we turn to formalizing the unlinkability of Scheme I. We basically rephrase the CPA-full-anonymity model defined by Boneh et al. [25], which is a slightly weaker version of the full-anonymity model given by Bellare et al. [28]. In this model, the adversary $\mathcal{A}$ is given a single public key. His goal is to determine which of two users is involved in an instance of the anonymous access control protocol. His success probability, $\mathrm{SUCC}_{\mathcal{A}}^{SchemeI}$, is defined as his probability of success in the game.

Definition (CPA-Full-Anonymity of Scheme I). An adversary $\mathcal{A}$ $(N, t, \mathrm{K}, \Gamma, \varepsilon)$-breaks the CPA-full-anonymity of the N-user Scheme I if $\mathcal{A}$ runs in time at most t, makes at most K queries to the hash function, issues at most Γ credential queries to the challenger, and $\mathrm{SUCC}_{\mathcal{A}}^{SchemeI}$ is at least ε.

Definition (Security of Scheme I). Scheme I is secure if no algorithm $(N, t, \mathrm{K}, \Gamma, \varepsilon)$-breaks its existential unforgeability and no algorithm $(N, t, \mathrm{K}, \Gamma, \varepsilon)$-breaks its CPA-full-anonymity.

Lemma 1.3. Proposed Scheme I achieves CPA-full-anonymity.

Corollary 1.3. Based on Theorem 1.1 and Lemma 1.3, proposed Scheme I is secure.

5.2 Scheme II

Here we present our second fast batch verifiable anonymous credential scheme (Scheme II). Unlike Scheme I, Scheme II works in the general system model and the security of Scheme II does not rely on the random oracle.

***PAKeyGen(·)*:** The PA calls $Setup(\cdot)$ according to the security parameter 1^k and chooses $a \in_R \mathbb{Z}_p$ and computes $A = g_2^{\,a} \in \mathbb{G}_2$.

The PA's public key is $pk^{PA} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e, A)$ and private key is $sk^{PA} = a$.

RHKeyGen(·): Given PA public key pk^{PA}, the resource holder RH_i that controls access rights R_{ij}, $j = 1, 2, \cdots, n_i$, executes the following:

2.a) For each R_{ij}, it chooses $b_{ij} \in_R \mathbb{Z}_p$ and computes $B_{ij} = g_2^{\,b_{ij}} \in \mathbb{G}_2$.

2.b) For each B_{ij}, it generates the signature of knowledge proof

$$\Sigma_{ij} = SKP\left\{(b_{ij}) : B_{ij} = g_2^{\,b_{ij}}\right\}.$$

The RH_i's access right public key is $pk_{ij}^{RH} = \left\{\left(R_{ij}, B_{ij}, \Sigma_{ij}\right)\right\}$ and the private key is $sk_{ij}^{RH} = b_{ij}$.

***UserEnroll(·)*:** In order to obtain a pseudonym, a user who has trustworthy identity U_u carries out the following with the PA:

3.a) The user sends its identity U_u to the PA and the PA queries its database for stored key z_u of U_u. Iff it does not find a match, the PA selects $z_u \in_R \mathbb{Z}_p$ for U_u and stores (U_u, z_u) in its database.

3.b) PA computes a BB signature [17] $t_u = g_1^{1/(a+z_u)}$, and sends user key z_u and pseudonym t_u to U_u.

3.c) User U_u verifies that $e(t_u, A \cdot g_2^{z}) = e(g_1, g_2)$.

***GCred(·)*:** In order to be granted a credential for access right R_{ij}, the user U_u carries out the following access-right-granting protocol with the resource holder RH_i that controls access right R_{ij}:

4.a) User U_u interacts with the RH_i for $KP\left\{(z):\ e(t_u, A) / e(g_1, g_2) = e(t_u^{-z}, g_2)\right\}$.

4.b) If $KP\left\{(z):\ e(t_u, A) / e(g_1, g_2) = e(t_u^{-z}, g_2)\right\} = \text{TRUE}$, the RH_i computes $v_{ij} = t_u^{b_{ij}}$ and sends v_{ij} to the user.

4.c) The user verifies that $e(t_u, B_{ij}) = e(v_{ij}, g_2)$ holds and stores v_{ij} as his credential for access right R_{ij}.

Fast Batch Verification of Anonymous Credentials: Suppose the user U_u has been granted pseudonym t_u and credentials for access rights $\left\{R_{ij}\right\}$ whose corresponding access-right public keys are $\left\{B_{ij}\right\}$. Without loss of generality, suppose one subset $\left\{r_j\right\} \subseteq \left\{R_{ij}\right\}$ of the user's access rights matches one of the RG's policies $Pol = \left(\left\{r_j\right\}, \mathrm{B} = \prod_j B_j, \ldots\right)$, where B_j is the access-right public key that corresponds to resource r_j. Since $r_j \in \left\{R_{ij}\right\}$, we have that $B_j \in \left\{B_{ij}\right\}$. Let v_j denote the user's credential that corresponds to access right r_j.

Define $VCred(\cdot)$ as: $e(t_u, A \cdot g_2^{z}) \overset{?}{=} e(g_1, g_2)$ and $e(t_u, B_j) \overset{?}{=} e(v_j, g_2)$. In order to prove to the RG that he meets policy Pol, the user U_u carries out the following anonymous access control protocol with the RG:

5.a) The user U_u selects $\tau \in_R \mathbb{Z}_p$, computes pseudonym $\mathrm{T} = t_u^{\tau}$ and batch verifiable anonymous credential $\mathrm{V} = \mathrm{v}^{\tau} = \left(\prod_j v_j\right)^{\tau}$.

5.b) User U_u notifies the RG of the matching of policy Pol and sends T, V to RG.

5.c) The RG interacts with the user U_a for

$$KP\left\{(t_u,\tau,z):\ e(\mathrm{T},A)=e(g_1^{\tau}\cdot\mathrm{T}^{-z},g_2)\wedge\mathrm{T}=t_u^{\tau}\right\}$$

utilizing Protocol 1.

5.d) If $KP\left\{(t_u,\tau,z):\ e(\mathrm{T},A)=e(g_1^{\tau}\cdot\mathrm{T}^{-z},g_2)\wedge\mathrm{T}=t_u^{\tau}\right\}=\mathrm{TRUE}$, the RG next verifies that $e(\mathrm{T},\mathrm{B})\overset{?}{=}e(\mathrm{V},g_2)$. If the congruence holds, the RG is convinced.

5.2.1 Scheme II Security

We formalize the existential unforgeability of Scheme II as an adaptive chosen-key and adaptive chosen-access-right game. In this model, the adversary $\mathcal{A}$ is given the PA public key and an access-right public key. His goal is the existential forgery of a pseudonym and a batch verifiable anonymous credential. The adversary's advantage, $\mathrm{ADV}_{\mathcal{A}}^{SchemeII}$, is defined as his probability of success in the game.

Definition (Existential Unforgeability of Scheme II). An adversary $\mathcal{A}$ (N,t,Γ,ε)-breaks the existential unforgeability of the N-user Scheme II in the adaptive chosen-key and adaptive chosen-access-right model if $\mathcal{A}$ runs in time at most t, issues at most Γ queries to the challenger, and $\mathrm{ADV}_{\mathcal{A}}^{SchemeII}$ is at least ε.

Theorem 2.1. Proposed Scheme II is secure against existential forgery under q-SDH assumption.

Corollary 2.1. Proposed Scheme II is a batch verification scheme for anonymous credentials.

Corollary 2.2. Based on Claim 1, proposed Scheme II is a *fast* batch verification scheme for anonymous credentials.

Now we turn to formalizing the unlinkability of Scheme II, again in the CPA-full-anonymity model. In this model, the adversary $\mathcal{A}$ is given the PA public key and some access-right public keys. His goal is to determine which of two users is involved in an instance of the anonymous access control protocol. His success probability, $\mathrm{SUCC}_{\mathcal{A}}^{SchemeII}$, is defined as his probability of success in the game.

Definition (CPA-Full-Anonymity of Scheme II). An adversary $\mathcal{A}$ (N,t,Γ,ε)-breaks the CPA-full-anonymity of the N-user Scheme II if $\mathcal{A}$ runs in time at most t, issues at most Γ queries to the challenger, and $\mathrm{SUCC}_{\mathcal{A}}^{SchemeII}$ is at least ε.

Definition (Security of Scheme II). Scheme II is secure if no algorithm (N,t,Γ,ε)-breaks its existential unforgeability and no algorithm (N,t,Γ,ε)-breaks its CPA-full-anonymity.

Lemma 2.3. Proposed Scheme II achieves CPA-full-anonymity.

Corollary 2.3. Based on Theorem 2.1 and Lemma 2.3, proposed Scheme II is secure.

6 Revocation

In this section, we will present a fast batch verifiable as well as fast fine-grained revocable scheme (Scheme III) that is modified from Scheme II.

6.1 Fast Fine-Grained Revocation

In addition to the five algorithms described in Section 4, Scheme III also requires the $Revoke(\cdot)$ algorithm below.

$Revoke(\cdot)$. This deterministic algorithm takes as input sk^{PA}, R_{ij}, pk_{ij}^{RH}, x_l, $Cred_{ijl}$, and $\tilde{x}$ for which the credential for R_{ij} needs revocation, and returns the updated $pk_{ij}^{RH\prime}$ and credential $Cred_{ijl}{}'$.

Definition (Fast Batch Verifiable and Fast Fine-grained Revocable Anonymous Credentials). A batch verifiable and fine-grained revocable anonymous credential scheme is said to achieve fast batch verification and fast fine-grained revocation properties if, for all $\hat{\mathcal{R}} \subseteq \tilde{\mathcal{R}} \subseteq \left\{R_{ij}\right\}$ where $\hat{\mathcal{R}}$ is the set of access rights whose corresponding credentials purportedly have not been revoked, the number of expensive cryptographic computations spent and the pieces of data generated by $KP\left\{(nym_l, x_l, \left\{Cred_{ijl}\right\}) : VCred(pk^{PA}, nym_l, x_l, \tilde{\mathcal{R}}, \left\{pk_{ij}^{RH}\right\}, \left\{Cred_{ijl}\right\})\right\} = \text{TRUE}$ are independent of $\left|\tilde{\mathcal{R}}\right|$ and $\left|\hat{\mathcal{R}}\right|$, where $VCred(\cdot)$ will return FALSE if any one credential for $\hat{\mathcal{R}}$ has been revoked.

In order to support fast fine-grained revocation, Scheme II's procedures for $RHKeyGen(\cdot)$, $GCred(\cdot)$, and anonymous access control protocol need to be slightly modified, as depicted below.

***RHKeyGen(·)*:** Given PA public key pk^{PA}, the resource holder RH_i that controls access rights R_{ij}, $j = 1, 2, \cdots, n_i$, executes the following:

2.a) For each R_{ij}, it chooses $b_{ij} \in_R \mathbb{Z}_p$ and computes $B_{ij} = g_2^{b_{ij}} \in \mathbb{G}_2$.

2.b) For each R_{ij}, it computes access right revocation data $h_{ij} = g_1^{b_{ij}} \in \mathbb{G}_1$ and initializes revocation list $RL_{ij} = \left\{(h_{ij}, \Delta)\right\}$, where Δ denotes that the two-tuple (h_{ij}, Δ) is the first row in RL_{ij}, i.e., no revocation happens yet.

2.c) For each B_{ij} and h_{ij}, it generates a signature of knowledge proof

$$\Sigma_{ij} = SKP\left\{(b_{ij}) : B_{ij} = g_2^{\,b_{ij}} \wedge h_{ij} = g_1^{\,b_{ij}}\right\}.$$

The RH_i 's public key is $pk_{ij}^{RH} = \left\{\left(R_{ij}, B_{ij}, \Sigma_{ij}, RL_{ij}\right)\right\}$ and the private key is $sk_{ij}^{RH} = b_{ij}$.

***GCred(·)*:** In order to be granted access right R_{ij}, the user U_u carries out the following access-right-granting protocol with the resource holder RH_i that controls access right R_{ij}:

4.a) User U_u interacts with the RH_i for $KP\left\{(z) : \; e(t_u, A) / e(g_1, g_2) = e(t_u^{\,-z}, g_2)\right\}$.

4.b) If $KP\left\{(z) : \; e(t_u, A) / e(g_1, g_2) = e(t_u^{\,-z}, g_2)\right\} = \text{TRUE}$, the RH_i computes $v_{ij} = t_u^{\,b_{ij}}$ and sends v_{ij} to the user.

4.c) The user U_u verifies that $e(t_u, B_{ij}) = e(v_{ij}, g_2)$ holds, and stores v_{ij} as his credential and $w_{ij} = v_{ij}$ as his validity data for access right R_{ij}.

***Revoke(·)*:** Given a misbehaving user $\tilde{U}$ that has user key $\tilde{z}$, in order to revoke his credential for access right R_{ij}, PA needs to do the following:

6.a) PA retrieves revocation data h_{ij} from the last (latest) row of RL_{ij}.

6.b) PA computes $\tilde{h}_{ij} = h_{ij}^{\,1/(a+\tilde{z})}$ and appends $(\tilde{h}_{ij}, \tilde{z})$ to RL_{ij}.

Consider user U_u that has user key z. User U_u has credential v_{ij} and validity data w_{ij} for access right R_{ij} as well. As a consequence of user $\tilde{U}$'s credential for access right R_{ij} being revoked, user U_u needs to execute the following to update his credential:

6.c) The user U_u computes $\tilde{w}_{ij} = (\tilde{h}_{ij} / w_{ij})^{1/(z-\tilde{z})}$ and updates his credential for access right R_{ij} to $(v_{ij}, \tilde{w}_{ij})$.

Fast Batch Verification & Fast Fine-Grained Revocation of Anonymous Credentials: Suppose the user U_u has been granted credentials for access rights $\left\{R_{ij}\right\}$ whose corresponding access-right public keys are $\left\{B_{ij}\right\}$. Without loss of generality, suppose one subset $\left\{r_j\right\} \subseteq \left\{R_{ij}\right\}$ of the user's access rights matches one of the RG's policies $Pol = \left(\left\{r_j\right\}, \mathrm{B} = \prod_j B_j, \ldots\right)$, where B_j is the access-right public key that corresponds to resource r_j. Since $r_j \in \left\{R_{ij}\right\}$, we have that $B_j \in \left\{B_{ij}\right\}$. Let v_{ij} denote the user's credential for credential access right r_j. Let w_{ij} denote the user's current validity data, i.e., data that has been updated to include the latest revocation, on access right r_j.

The user U_u wants to prove to the RG that he meets policy Pol. Whereas, the RG is curious about whether the user's access rights on $\{r_k\} \subseteq \{r_j\}$ have been revoked. Let h_k denote the access right revocation data that is retrieved from the last row of RL_k.

Define $VCred(\cdot)$ as $e(t_u, A \cdot g_2^{\ z}) \overset{?}{=} e(g_1, g_2)$, $e(t_u, B_j) \overset{?}{=} e(v_j, g_2)$, and $e(w_k, A \cdot g_2^{\ z}) \overset{?}{=} e(h_k, g_2)$. In order to convince the RG, the user U_u carries out the following anonymous access control protocol with the RG:

5.a) The user U_u selects $\tau \in_R \mathbb{Z}_p$, and computes pseudonym $\mathrm{T} = t_u^{\ \tau}$ and batch verifiable anonymous credential $\mathrm{V} = \mathrm{v}^{\tau} = \left(\prod_j v_j\right)^{\tau}$. In addition, the user computes fine-grained validity data $\mathrm{W} = \mathrm{w}^{\tau} = \left(\prod_k w_k\right)^{\tau}$.

5.b) The user U_u notifies the RG of the matching of policy Pol and sends T, V, and W to the RG.

5.c) Utilizing a natural extension of Protocol 1, the RG interacts with the user U_u for

$$KP\left\{\begin{aligned}(t_u, \mathrm{w}, \tau, z):\ & e(\mathrm{T}, A) = e(g_1^{\ \tau} \cdot \mathrm{T}^{-z}, g_2) \wedge \mathrm{T} = t_u^{\ \tau} \\ & \wedge\, e(\mathrm{W}, A) = e((\prod_k h_k)^{\tau} \cdot \mathrm{W}^{-z}, g_2) \wedge \mathrm{W} = \mathrm{w}^{\tau}\end{aligned}\right\}.$$

5.d) If the RG accepts the above knowledge proof, it further verifies that $e(\mathrm{T}, \mathrm{B}) \overset{?}{=} e(\mathrm{V}, g_2)$. If the congruence holds, the RG is convinced.

6.1.1 Security of Fine-Grained Revocation

Note that Scheme III exactly reuses the steps in Scheme II to achieve fast batch verification of anonymous credentials and that Protocol 1 is witness indistinguishable [33]. Therefore, Scheme III should be secure with respect to existential unforgeability and CPA-full-anonymity, as long as the fine-grained validity data is existentially unforgeable.

We formalize the existential unforgeability of the validity data as an adaptive chosen-key and adaptive chosen-access-right game. In this model, the adversary $\mathcal{A}$ is given the PA public key, an access-right public key, and access right revocation data. His goal is the existential forgery of the validity data. The adversary's advantage, $\mathrm{ADV}_{\mathcal{A}}^{VALID}$, is defined as his probability of success in the game.

Definition (Existential Unforgeability). An adversary $\mathcal{A}$ $(N, t, \Gamma, \varepsilon)$-breaks the existential unforgeability of validity data of the N-user Scheme III in the adaptive chosen-key and adaptive chosen-access-right model if $\mathcal{A}$ runs in time at most t, issues at most Γ queries to the challenger, and $\mathrm{ADV}_{\mathcal{A}}^{VALID}$ is at least ε.

Theorem 3.1. Proposed Scheme III is secure against existential forgery of validity data under q-SDH assumption.

Corollary 3.1. Based on Claim 1, proposed Scheme III attains *fast* fine-grained revocation of anonymous credentials.

7 Conclusions and Future Work

Three constructions of *fast* batch verifiable anonymous credential schemes were presented. The first scheme achieves the *fast* batch verifiable property in the random oracle model. The second and third schemes achieve the *fast* batch verifiable property in the standard model. The third scheme in addition achieves the *fast* fine-grained revocable property.

To attain the *fast* properties, our three schemes require linear storage consumption for the credentials and the credentials cannot be arbitrary statements. It is desirable to see anonymous credential scheme with *fast* properties that overcomes these two limitations.

According to the findings of [24] and [34], our three schemes after further modifications may be able to conduct batch verification of anonymous credentials from different users. But such modifications cannot be *fast* if we require the number of expensive cryptographic computations being independent of the number of users. It is thus very interesting to find a *fast* scheme for this usage scenario.

Acknowledgements

The author is grateful to Tomoyuki Fujita, Min-Yu Hsueh, and Toshikazu Fukushima for pointing to this research and for support to this research. The author thanks Yu-Liang Zheng, Yong Ding, Hao Lei, Ye Tian, and Li-Ming Wang for valuable discussions during the development of the ideas herein presented. The author also thanks the anonymous reviewers, whose comments and suggestions help to improve the quality of this paper and stimulate new thoughts. The author is particularly grateful to one reviewer for pointing out a mistake in an early version of this paper.

References

1. Verheul, E.R.: Self-Blindable Credential Certificates from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 533–551. Springer, Heidelberg (2001)
2. Westhoff, D., Lamparter, B.: Charging Related Mobile Device Authentication. In: Advanced Internet Charging and QoS Technologies (ICQT 2001), pp. 129–135 (2001), ISBN 3-85403-157-2

3. Odlyzko, A.: Privacy, Economics, and Price Discrimination on the Internet. In: 5th International Conference on Electronic Commerce, pp. 355–366. ACM Press, New York (2003)
4. 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)
5. Chaum, D.: Security without Identification: Transaction Systems to Make Big Brother Obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
6. Chaum, D., Evertst, J.H.: A Secure and Privacy-protecting Protocol for Transmitting Personal Information Between Organizations. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 118–167. Springer, Heidelberg (1987)
7. Chen, L.: Access with Pseudonyms. In: Dawson, E.P., Golić, J.D. (eds.) Cryptography: Policy and Algorithms 1995. LNCS, vol. 1029, pp. 232–243. Springer, Heidelberg (1996)
8. Lysyanskaya, A., Rivest, R., Sahai, A., Wolf, S.: Pseudonym Systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
9. Holt, J.E., Seamons, K.E.: Selective Disclosure Credential Sets. Report 2002/151, Cryptology ePrint Archive (2002)
10. Persiano, P., Visconti, I.: An Anonymous Credential System and a Privacy-Aware PKI. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 27–38. Springer, Heidelberg (2003)
11. Persiano, P., Visconti, I.: An Efficient and Usable Multi-show Non-transferable Anonymous Credential System. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 196–221. Springer, Heidelberg (2004)
12. 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)
13. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Non-Interactive Anonymous Credentials. Report 2007/384, Cryptology ePrint Archive (2007)
14. Akagi, N., Manabe, Y., Okamoto, T.: An Efficient Anonymous Credential System. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143. Springer, Heidelberg (2008)
15. Ateniese, G., Song, D., Tsudik, G.: Quasi-efficient Revocation of Group Signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
16. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch Verification of Short Signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)
17. 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)
18. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solution to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
19. Chaum, D., Evertse, J.H., 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)
20. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
21. Qin, B., Wu, Q., Susilo, W., Mu, Y.: Group Decryption. Report 2007/017, Cryptology ePrint Archive (2007)

22. Okamoto, T.: Provable Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
23. Bellare, M., Garay, J.A., Rabin, T.: Fast Batch Verification for Modular Exponentiation and Digital Signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
24. Ferrara, A.L., Green, M., Hohenberger, S., Pedersen, M.O.: On the Practicality of Short Signature Batch Verification. Report 2008/015, Cryptology ePrint Archive (2008)
25. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
26. Miyaji, A., Nakabayashi, M., Takano, S.: New Explicit Conditions of Elliptic Curves for FR-reduction. IEICE Trans. Fundamentals E84-A(5), 1234–1243 (2001)
27. MIRACL, Multi-precision Integer and Rational Arithmetic C Library, `http://www.shamus.ie`
28. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
29. 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)
30. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–77. Springer, Heidelberg (2002)
31. Nguyen, L.: Accumulators from Bilinear Pairings and Applications to ID-based Ring Signatures and Group Membership Revocation. Report 2005/123, Cryptology ePrint Archive (2005)
32. Ateniese, G., Camenisch, J., Hohenberger, S., Medeiros, B.: Practical Group Signatures without Random Oracles. Report 2005/385, Cryptology ePrint Archive (2005)
33. Feige, U., Shamir, A.: Witness Indistinguishable and Witness Hiding Protocols. In: 22nd ACM Symposium on Theory of Computing, pp. 416–426. ACM Press, New York (1990)
34. Peng, K., Boyd, C., Dawson, E.: Batch Zero-Knowledge Proof and Verification and Its Applications. ACM Transactions on Information and System Security, Article 6 10(2), 1–28 (2007)

Quantifying Timing Leaks and Cost Optimisation

Alessandra Di Pierro[1], Chris Hankin[2], and Herbert Wiklicky[2]

[1] University of Verona, Ca' Vignal 2 - Strada le Grazie 15 I-37134 Verona, Italy
[2] Imperial College London, 180 Queen's Gate London SW7 2AZ, UK

Abstract. We develop a new notion of security against timing attacks where the attacker is able to simultaneously observe the execution time of a program and the probability of the values of low variables. We then show how to measure the security of a program with respect to this notion via a computable estimate of the timing leakage and use this estimate for cost optimisation.

1 Introduction

Early work on language-based security, such as Volpano and Smith's type systems [1], precluded the use of high security variables to affect control flow. Specifically, the conditions in if-commands and while-commands were restricted to using only low security information. If this restriction is weakened, it opens up the possibility that high security data may be leaked through the different timing behaviour of alternative control paths. This kind of leakage of information is said to form a *covert timing channel* and is a serious threat to the security of programs (cf. e.g. [2]).

We develop a new notion of security against timing attacks where the attacker is able to simultaneously observe the execution time of a (probabilistic) program and the probability of the values of low variables. This notion is a non-trivial extension of similar ideas for deterministic programs [3] which also covers attacks based on the combined observation of time and low variables. This earlier work presents an approach which, having identified a covert timing channel, provides a program transformation which neutralises the channel.

We start by introducing a semantic model of timed probabilistic transition systems. Our approach is based on modelling programs essentially as Markov Chains (MC) where the stochastic behaviour is determined by a joint distribution on both the values assigned to the program's variables and the time it takes to the program to perform a given command. This is very different from other approaches in the area of automata theory which are also dealing with both time and probability. In this area the timed automata constitute a well-established model [4]. These automata have been extended with probability and used in model-checking for the verification of probabilistic timed temporal logic properties of real-time systems. The resulting model is essentially a Markov Decision Process (MDP) where rewards are interpreted as time durations. In particular,

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 81–96, 2008.

the presence of non-determinism makes MDP models not very appropriate as a base of our quantitative analysis aiming at measuring timing leaks. We next present a concrete programming language with a timed probabilistic transition system as its execution model. This language is based on the language studied in [3] but is extended with a probabilistic choice construct – whilst this may not play a role in user programs, it has an essential role in our program transformation. In order to determine and quantify the security of systems and the effectiveness of potential counter-measures against timing attacks we then discuss an approximate notion of timed bisimilarity and construct an algorithm for computing a quantitative estimate of the vulnerability of a system against timing attacks; this is given in terms of the mismatch between the actual transition probabilities and those of an ideal perfectly confined program. Finally, we present a probabilistic variation of Agat's padding algorithm which we use to illustrate – via an example – a technique for formally analysing the trade-off between security costs and protection.

2 The Model

We introduce a general model for the semantics of programs where time and probability are explicitly introduced in order to keep track of both the probabilistic evolution of the program/system state and its running time.

The scenario we have in mind is that of a multilevel security system and an attacker who can observe the system by looking at the values of its public variables and the time it takes to perform a given operation, or before terminating, or other similar properties related to its timing behaviour. In order to keep the model simple, we assume that the time to execute a statement is constant and that there is no distinction between any 'local' and 'global' clocks. In a more realistic model, one has – of course – to take into account also that the execution speed might differ depending on which other process is running on the same system and/or delays due to uncontrollable events in the communication infrastructure, i.e. network.

Our reference model is the timed probabilistic transition system we define below. The intuitive idea is that of a probabilistic transition system (similar to those defined in all generality in [5]) where transition probabilities are defined by a joint distribution of two random variables representing the variable updates and time, respectively.

Let us consider a finite set X, and let $\mathbf{Dist}(X)$ denote the set of all *probability distributions* on X, that is the set of all functions $\pi : X \rightarrow [0, 1]$, such that $\sum_{x \in X} \pi(x) = 1$. We often represent these functions as sets of tuples $\{\langle x, \pi(x) \rangle\}_{x \in X}$. If the set X is presented as a Cartesian product, i.e. $X = X_1 \times X_2$, then we refer to a distribution on X also as a *joint distribution* on X_1 and X_2. A joint distribution associates to each pair (x_1, x_2), with $x_1 \in X_1, x_2 \in X_2$ the probability $\pi(x_1, x_2)$. It is important to point out that, in general, it is not possible to define any joint distribution on $X_1 \times X_2$ as a 'product' of distributions on X_1 and X_2, i.e. for a given joint distribution π on $X = X_1 \times X_2$ it is, in general, not possible to find distributions π_1 and π_2 on X_1 and X_2 such that for all $(x_1, x_2) \in X_1 \times X_2$ we have $\pi(x_1, x_2) = \pi_1(x_1)\pi_2(x_2)$. In the special cases where a joint distribution π can be expressed in this way, as a 'product', we say that the distributions π_1 and π_2 are *independent* (cf. e.g. [6]).

2.1 Timed Probabilistic Transition Systems

The execution model of programs which we will use in the following is that of a labelled transition system; more precisely, we will consider probabilistic transition systems (PTS). We will put labels on transitions as well as states; the former will have "times" associated with them while the latter will be labelled by uninterpreted entities which are intended to represent the values of (low security) variables, i.e. the computational state during the execution of a program. We will not specify what kind of "time labels" we use – e.g. whether we have a discrete or continuous time model – we just assume that time labels are taken from a finite set $\mathbb{T} \subseteq \mathbb{R}^+$ of positive real numbers. The "state labels" will be taken from an abstract set which we denote by $\mathbb{L}$.

Definition 1. *We define a* timed Probabilistic Transition System with labelled states*, or tPTS, as a triple* $(S, \longrightarrow, \lambda)$*, with* S *a finite set of* states*,* $\longrightarrow \subseteq S \times \mathbb{T} \times [0, 1] \times S$ *a probabilistic transition relation, and* $\lambda : S \to \mathbb{L}$ *a state labelling function.*

We denote by $s_1 \xrightarrow{p:t} s_2$ the fact that $(s_1, p, t, s_2) \in \longrightarrow$ with $s_1, s_2 \in S$, $p \in [0, 1]$ and $t \in \mathbb{T}$. In a general tPTS we can have *non-determinism* in the sense that for two states s_1 and s_2 we may have $s_1 \xrightarrow{1:t_1} s_2$ and $s_1 \xrightarrow{1:t_2} s_2$, which would suggest that it is possible to make a transition from s_1 to s_2 in different times (t_1 and t_2) and probability 1, i.e. certainly. In order to eliminate non-determinism we will consider in this paper only tPTS's which are subject to the following conditions: (i) for all $s \in S$ we have $\sum_{(s,p_i,t_j,s_k)\in\longrightarrow} p_i = 1$, and (ii) for all $t \in T$ there is *at most one* tuple $(s_1, t, p, s_2) \in \longrightarrow$.

The first condition means that we consider here a *purely probabilistic* or *generative* execution model. The second condition allows us to associate a unique probability to every transition time between two states, i.e. a triple (s_1, t, s_2); this means that we can define a function $\pi : S \times \mathbb{T} \times S \to [0, 1]$ such that $s_1 \xrightarrow{p:t} s_2$ iff $\pi(s_1, t, p_2) = p$. Note however, that it is still possible to have differently timed transitions between states, i.e. it is possible to have $(s_1, t_1, p_2, s_2) \in \longrightarrow$ and $(s_1, t_2, p_2, s_2) \in \longrightarrow$ with $t_1 \neq t_2$. If for all $s_1, s_2 \in S$ there exists at most one $(s_1, t, p, s_2) \in \longrightarrow$, we can also represent a timed Probabilistic Transition System with labelled states as a quadruple $(S, \longrightarrow, \tau, \lambda)$ with $\tau : S \times S \to [0, 1] \times \mathbb{T}$, a timing function. Thus, to any two states s_1 and s_2 we associate a unique transition time t_{s_1,s_2} and probability p_{s_1,s_2}.

Definition 2. *Consider a tPTS* $(S, \longrightarrow, \lambda)$ *and an* initial state $s_0 \in S$*. An* execution sequence *or* trace *starting in* s_0 *is a sequence* $(s_0, s_1, \ldots)$ *such that* $s_i \xrightarrow{p_i:t_i} s_{i+1}$*, for all* $i = 0, 1, 2, \ldots$.

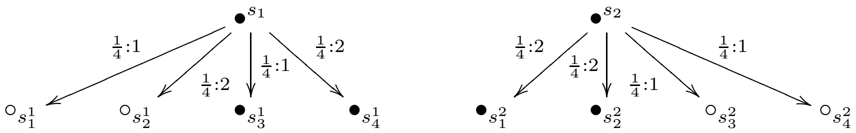
We associate, in the obvious way, to an execution sequence $\sigma = (s_0, s_1, \ldots)$ three more sequences: (i) the transition probability sequence: $(p_1, p_2, \ldots)$, (ii) a time stamp sequence: $(t_1, t_2, \ldots)$, and (iii) a state label sequence: $(\lambda(s_o), \lambda(s_1), \ldots)$.

Even for a tPTS with a finite number of states it is possible to have infinite execution sequences. It is thus, in general, necessary to consider measure theoretic notions in order to define a mathematically sound model for the possible behaviours of a tPTS. However, as long as we consider only terminating systems, i.e. finite traces, things are somewhat simpler. In particular probability distributions can replace measures as they are equivalent in this case. In this paper we restrict our attention to terminating traces and probability distributions. This allows us to define for every finite execution sequence $\sigma = (s_0, s_1, \ldots)$ its *running time* as $\tau(\sigma) = \sum t_i$, and its *execution probability* as $\pi(\sigma) = \prod t_i$. We will also associate to every state s_0 its *execution tree*, i.e. the collection of all execution sequences starting in s_0.

2.2 Observing tPTS's

In Section 3 we will present an operational semantics of a simple imperative programming language, pWhile, via a tPTS. Based on this model we will then investigate the vulnerability against attackers which are able to observe (i) the time, and (ii) the state labels, i.e. the low variables. In this setting we will argue that the combined observation of time and low variables is more powerful than the observation of time and low variables separately.

Example 1. In order to illustrate the role of joint distributions in the observation of timed PTS's let us consider the following simple systems.



We assume that the attacker can observe the execution times and that he/she is also able to (partially) distinguish (the final) states. In our example we assume that the states depicted as • and ◦ form two classes which the attacker can identify (e.g. because the • and ◦ states have the same values for low variables). The question now is whether this information allows the attacker to distinguish the two tPTS's.

If we consider the information obtained by observing the running time, we see that both systems exhibit the same time behaviour corresponding to the distribution $\{\langle 1, \frac{1}{2}\rangle, \{\langle 2, \frac{1}{2}\rangle\}$ over $\mathbb{T} = \{1, 2\}$. The same is true in the case where the information is obtained by inspecting the final states: we have the distributions $\{\langle \bullet, \frac{1}{2}\rangle, \{\langle \circ, \frac{1}{2}\rangle\}$ over $\mathbb{L} = \{\bullet, \circ\}$ for both systems.

However, considering that the attacker can observe running time and labels simultaneously, we see that the system on the rhs always runs for 2 time steps iff it ends up in a • state and 1 time step iff it ends up in a ◦ state. In the system on the lhs there is no such *correlation* between running time and final state. The difference between the two systems, which allows an attacker to distinguish them, is reflected in the joint distributions over $\mathbb{T} \times \mathbb{L}$. These are $\chi_1(t,l) = \frac{1}{4}$ for all $t = 1, 2$ and $l = \bullet, \circ$; and $\chi_2(1,\circ) = \frac{1}{2} = \chi_2(2,\bullet)$ and $\chi_2(t,l) = 0$ otherwise. Note that while χ_1 is the product of two *independent* probability distributions on $\mathbb{T}$ and $\mathbb{L}$ it is not possible to represent χ_2 in the same way.

3 An Imperative Language

We consider a language similar to that used in [3] with the addition of a probabilistic choice construct. The syntax of the language, which we call pWhile, is as follows:

$$\begin{array}{lrl}
\text{Operators:} & op & ::= + \mid * \mid - \mid = \mid != \mid < \mid <= \\
\text{Expressions:} & e & ::= v \mid x \mid e\ op\ e \\
\text{Commands:} & C, D & ::= x := e \mid \textbf{skipAsn}\ x\ e \mid \textbf{if}\ (e)\ \textbf{then}\ C\ \textbf{else}\ D \mid \textbf{skipIf}\ e\ C \\
& & \mid \textbf{while}\ (e)\ \textbf{do}\ C \mid C; D \mid \textbf{choose}^p\ C\ \textbf{or}\ D \\
\text{Basic Values:} & v & ::= n \mid \textbf{true} \mid \textbf{false}
\end{array}$$

The probabilistic choice, **choose**^{p} C **or** D, is used in an essential way in the program transformation presented later. We also keep the language of types in [3], although in a simplified form (with L ≤ H and $s \le s$):

$$\begin{array}{lrl}
\text{Security levels:} & s & ::= \mathrm{L} \mid \mathrm{H} \\
\text{Base types:} & \overline{\tau} & ::= \textbf{Int} \mid \textbf{Bool} \\
\text{Security types:} & \tau & ::= \overline{\tau}_s
\end{array}
\quad \text{and sub-typing: } \frac{s_1 \le s_2}{\overline{\tau}_{s_1} \le \overline{\tau}_{s_2}}.$$

We will indicate by E the state of a computation and denote by E_L its restriction to low variables, i.e. a state which is defined as E for all the low variables for which E is defined, and is undefined otherwise. We say that two configurations $\langle E \mid C\rangle$ and $\langle E' \mid C'\rangle$ are *low equivalent* if and only if $E_L = E'_L$ and we indicate this by $\langle E \mid C\rangle =_L \langle E' \mid C'\rangle$. In the following we will sometimes use for configurations the shorthand notation $c, c_1, c_2, \dots, c', c'_1, \dots$. We will also denote by **Conf** the set of all configurations.

The big step semantics of expressions and the small-step semantics of commands are essentially the same as those in [3]. The only difference is the rule for probabilistic choice which we have added to the original semantics. We refer to the full version of this paper [7] for a complete description of this semantics and we only report here on the probabilistic choice rule(s) (Choose):

$$\frac{}{\langle E \mid \textbf{choose}^p\ C\ \textbf{or}\ D\rangle \xrightarrow{p:t_{ch}} \langle E \mid C\rangle} \qquad \frac{}{\langle E \mid \textbf{choose}^p\ C\ \textbf{or}\ D\rangle \xrightarrow{(1-p):t_{ch}} \langle E \mid D\rangle}$$

In this rule, t_{ch} indicates the time it takes to execute a choice command. In general, we will use the time labels $t_.$ to represent the time it takes to perform certain operations: t_x is the time to store a variable, t_e is the time it takes to evaluate an expression, t_{asn} represents the time to perform an assignment, t_{br} is the time required for a branching step, and t_{ch} is the time to perform a probabilistic choice.

The rule above states that the execution of a probabilistic choice construct leads, after a time t_{ch}, to a state where either the command C or the command D is to be executed with probability p or $1-p$, respectively. This rule together with the standard transition rules for the other constructs of the language define a tPTS for our pWhile language according to Definition 1. In this tPTS, the state labels are given by the environment, i.e. $\lambda(\langle E \mid C\rangle) = E$.

3.1 Abstract Semantics

According to the notion of security we consider in this paper, an observer or attacker can only observe the changes in low variables. Therefore, we can simplify the semantics by ‘collapsing’ the execution tree in such a way that execution steps during which the value of all low variables is unchanged are combined into one single step. We call an execution sequence σ *deterministic* if $\pi(\sigma) = 1$, and we call it *low stable* if $\lambda(s_i)|_L = l$ for all $s_i \in \sigma$. The empty path (of length zero) is by definition deterministic and low stable. An execution sequence is *maximal deterministic/low stable* if it is not a proper sub-sequence of another deterministic/low stable path.

Definition 3. *The collapsed transition relation* $\langle E_1 \mid C_1\rangle \overset{p:T}{\Longrightarrow} \langle E_2 \mid C_2\rangle$ *between two configurations is defined iff*

(i) *there exists a configuration* $\langle E'_1 \mid C'_1\rangle$ *such that* $\langle E_1 \mid C_1\rangle \xrightarrow{p:t} \langle E'_1 \mid C'_1\rangle$,

(ii) $\langle E'_1 \mid C'_1\rangle \xrightarrow{1:t_1} \ldots \langle E'_2 \mid C'_2\rangle \xrightarrow{1:t_n} \langle E_2 \mid C_2\rangle$ *is deterministic,*

(iii) $\langle E_1 \mid C_1\rangle \xrightarrow{p:t} \langle E'_1 \mid C'_1\rangle \ldots \xrightarrow{1:t_{n-1}} \langle E'_2 \mid C'_2\rangle$ *is maximal low stable,*

(iv) *and* $T = t + \sum_{i=1}^{n} t_i$.

4 Bisimulation and Timing Leaks

Observing the low variables and the running time separately is not the same as observing them together; a correlation between the two random variables (probability and time) has to be taken into account (cf. Section 2). A naive probabilistic extension of the Γ-bisimulation notion introduced in [3] might not take this into account. More precisely, this may happen if time and probability are treated as two independent aspects which are observed separately in a mutual exclusive way. According to such a notion an attacker must set up two different covert channels if he/she wants to exploit possible interference through both the probabilistic and the timing behaviour of the system. The notion of bisimulation we introduce here allows us to define a stronger security condition: an attacker must be able to distinguish the probabilities that two programs compute a given result in a given execution time. This is obviously different from being able to distinguish the probability distributions of the results *and* the running time.

Probabilistic bisimulation was first introduced in [8] and refers to an equivalence on probability distributions over the states of the processes. This latter equivalence is defined as a lifting of the bisimulation relation on the support sets of the distributions, namely the states themselves.

An equivalence relation $\sim \subseteq S \times S$ on S can be lifted to a relation $\sim^* \subseteq \mathbf{Dist}(S) \times \mathbf{Dist}(S)$ between probability distributions on S via (cf [5, Thm 1]): $\mu \sim^* \nu$ iff $\forall [s] \in S/_\sim : \mu([s]) = \nu([s])$. It follows that $\sim^*$ is also an equivalence relation ([5, Thm 3]). For any equivalence relation $\sim$ on the set **Conf** of configurations, we define the associated *low equivalence* relation $\sim_L$ by $c_1 \sim_L c_2$ if $c_1 \sim c_2$ and $c_1 =_L c_2$. Obviously $\sim_L$ is again an equivalence relation. We can lift a low equivalence $\sim_L$ to $(\sim_L)^*$ which we simply denote by $\sim_L^*$.

Definition 4. *Given a security typing Γ, a probabilistic time bisimilarity $\sim$ is the largest symmetric relation on configurations such that whenever $c_1 \sim c_2$, then $c_1 \Longrightarrow \chi_1$ implies that there exists χ_2 such that $c_2 \Longrightarrow \chi_2$ and $\chi_1 \sim_L^* \chi_2$.*

We say that two configurations are probabilistic time bisimilar or PT-bisimilar, $c_1 \sim c_2$, if there exists a probabilistic time bisimilarity relation in which they are related.

This definition generalises the one in [3] which only applies to deterministic transition systems. Note that there is a difference between $\sim_L^* = (\sim_L)^*$ and $(\sim^*)_L$; in fact, only the former is able to take into account the correlation between time and low variables, while the latter would be a straightforward generalisation of the time bisimulation in [3] which is unable to model such a correlation.

We now exploit the notion of bisimilarity introduced above in order to introduce a security property ensuring that a system is confined against any combined attacks based on both timing and probabilistic covert channels.

Definition 5. *A pWhile program P is* probabilistic time secure *or PT-secure if for any set of initial states E and E' such that $E_L = E'_L$, we have $\langle E, P\rangle \sim \langle E', P\rangle$.*

5 Computing Approximate Bisimulation

The papers [9,10] introduce an approximate version of bisimulation and confinement where the approximation can be used as a measure ε for the information leakage of the system under analysis. The quantity ε is formally defined in terms of the norm of a linear operator representing the partition induced by the 'minimal' bisimulation on the set of the states of a given system, i.e. the one minimising the observational difference between the system's components. We show here how to compute a non-trivial upper bound δ to ε by essentially exploiting the algorithmic solution proposed by Paige and Tarjan [11] for computing bisimulation equivalence. This was already adapted to PTS's in [12], where it was used for constructing a padding algorithm as part of a transformational approach to the timing leaks problem. In this approach the computational paths of a program are transformed so as to make it perfectly secure by eliminating any possible timing covert channel while preserving its I/O behaviour.

The algorithm we present here is an instantiation of that algorithm where the abstract labels are replaced by the statements in a concrete language (pWhile) and their execution times. Moreover, instead of transforming the execution trees, our algorithm accumulates the information about the difference between their transition probabilities and uses this information to compute an upper bound δ to the maximal information leakage of the given program.

5.1 Computing δ for PT-Bisimulation

Algorithm CompDelta in Table 1 describes the procedure for computing δ used inside the algorithm QLumping on the lhs of Table 1; this latter constructs a lumping (i.e. a PT-bisimulation equivalence) of two tPTS's T_1 and T_2 with states S_1 and S_2, respectively. QLumping follows the algorithmic paradigm for partition refinement introduced by Paige and Tarjan in [11]. The Paige-Tarjan algorithm constructs a partition of a state space Σ which is *stable* for a given transition relation $\rightarrow$. It is a well-known result that this partition corresponds to a bisimulation equivalence on the transition system $(\Sigma, \rightarrow)$. The refinement procedure used in the algorithm consists in *splitting* the blocks in a given partition P by replacing each block $B \in P$ with $B \cap preS$ and $B \setminus preS$, where $S \subseteq \Sigma$ and $pre(X) = \{s \in \Sigma \mid s \rightarrow x \text{ for some } x \in X\}$.

In order to check whether two execution trees T_1 and T_2 in our tPTS model are PT-bisimilar, we apply this refinement technique to the set of states formed by the disjoint union of the states in T_1 and T_2. The strategy of QLumping is as follows: the lumping procedure QLumping(T_1, T_2) works iteratively layer by layer starting from the leaves layer, and splits the blocks in the current partition restricted to the current layer. The procedure CompDelta(L_1, L_2) computes for each two layers L_1 and L_2, the maximal difference $\|\chi(s_1) - \chi(s_2)\|_\infty$ between the probabilities to get from states in $T_1 \cap L_1$ and $T_2 \cap L_1$, respectively, into states of layer L_2. In the original lumping procedure this determines a splitting of the

Table 1. Algorithms QLumping and CompDelta

```
procedure QLumping($T_1, T_2$)
  $\delta \leftarrow 0$, $n \leftarrow 0$ and $P \leftarrow \{S_1 \cup S_2\}$
  while $n \leq$ Height$(T_1 \oplus T_2)$ do
    $S \leftarrow \{B \cap$ CutOff$(T_1 \oplus T_2, n)) \mid B \in P\}$
    while $S \neq \emptyset$ do
      choose $B \in S$, $S \leftarrow S \setminus B$
      $P \leftarrow$ Splitting$(B, P)$
    end while
    $L_1 \leftarrow$ Layer$(T_1, n)$, $L_2 \leftarrow$ Layer$(T_2, n)$
    CompDelta$(L_1, L_2)$
    $n \leftarrow n + 1$
  end while
end procedure
```

```
procedure CompDelta($L_1, L_2$)
  while $L_1 \neq \emptyset$ do
    choose $s_1 \in L_1$, $L_1 \leftarrow L_1 \setminus s_1$
    $\beta \leftarrow \infty$
    $L \leftarrow L_2$
    while $L_2 \neq \emptyset$ do
      choose $s_2 \in L$, $L \leftarrow L \setminus s_2$
      $\beta \leftarrow \min(\beta, \|\chi(s_1) - \chi(s_2)\|_\infty)$
    end while
    $\delta \leftarrow \max(\delta, \beta)$
  end while
end procedure
```

states in layer L_1. This value is stored in a variable β and compared with the current value of a variable δ which contains the maximal difference up to that iteration. When the lumping algorithm terminates (that is when we have reached the root of the union tree), one of the following situations will occur: either the roots of T_1 and T_2 belong to the same class in the constructed partition (i.e. T_1 and T_2 are PT-bisimilar) or not. In the latter case δ will contain a maximal difference in the transition probabilities of the two processes which makes them non-bisimilar. This is therefore an estimate of the information leakage of the system. Note that, by construction, δ will be zero in the first case.

The strategy for constructing the lumping described above determines the *coarsest partition* of a set which is stable wrt a given relation, that is in our case the coarsest PT-bisimulation equivalence. Obviously, this does not necessarily coincide with the 'minimal' one corresponding to the quantity ε defined in [9]. Thus, δ will be in general only a safe approximation, namely an upper bound to the capacity of probabilistic timing covert channel defined by ε. The following proposition is therefore a corollary of Proposition 45 in [9] stating a similar assertion for ε-bisimulation.

Proposition 1. *P is PT-secure iff for any pair of initial configurations c_1, c_2 the corresponding execution trees T_1 and T_2 are such that* QLumping(T_1, T_2) *returns $\delta = 0$.*

5.2 A Weighted Version: δ'

The actual value of δ is determined by the way we compute the best match between the joint probability distributions $\chi(s_1)$ and $\chi(s_2)$ in line 8 of QLumping. In order to compute δ we use the *supremum norm*, $\|\cdot\|_\infty$, between two distributions, i.e. the largest absolute difference between corresponding entries in $\chi(s_1)$ and $\chi(s_2)$, respectively. In other words, we try to identify a class of states C (in the layer below) and a time interval t such that the probability of reaching this class in that time from s_1 differs maximally from the one for s_2.

One can argue that this is a fair approach as we treat all classes and time labels the same way. However, it might be useful to develop a measure which reflects the fact that certain times and classes are 'more similar' than others.

From the point of view of the attacker, such a measure would encode her/his ability in detecting similarity as given by the nature and the precision of the instruments he/she is actually using. For example, suppose it is possible to reach the same class C from s_1 and s_2 with different times t_1 and t_2, such that the corresponding probabilities determine δ (i.e. we have the maximal difference in this case). However, we might in certain circumstances also want to express the fact that t_1 and t_2 are more or less similar, e.g. for $t_1 = 10$ and $t_2 = 10.5$ we might want a smaller δ' than for $t_1 = 1$ and $t_2 = 100$. In terms of the attacker, this means that we make our estimate dependent on the actual power of the time detection instrument that he/she possesses.

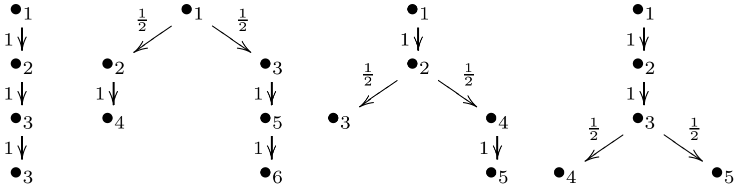
In order to incorporate similarity of times and/or classes we need to modify the way we determine the best match in line 8 of CompDelta(L_1, L_2). Instead of determining the norm between $\chi(s_1)$ and $\chi(s_2)$ we can compute a weighted version as:

$$\beta \leftarrow \min(\beta, \|\omega \cdot \chi(s_1) - \omega \cdot \chi(s_2)\|_\infty) = \min(\beta, \|\omega \cdot (\chi(s_1) - \chi \cdot (s_2))\|_\infty),$$

where ω re-scales the entries in $\chi(s_1)$ and $\chi(s_2)$ so as to reflect the relative importance of certain times and/or classes. Note that "$\cdot$" denotes here the component-wise and not the matrix multiplication: $(\omega \cdot \chi)_{tC} = \omega_{tC}\chi_{tC}$. If, for example, an attacker is not able to detect the absolute difference between times but can only measure multiplicities expressing approximative proportions, we could re-scale the χ's via $\omega_{tC} = \log(t)$.

In the following we will use a weighted version δ' which reflects the similarity of classes. The idea is to weight according to the "replaceability" of a class. To this purpose we associate to every class (in the layers below) a matching measure $\mu(C) = \min_{C \neq C'} \delta'(C, C')$, i.e. we determine the δ' between a (sub)tree with a root in the class C in question and all (sub)trees with roots in any of the other classes C'. We can take any representative of the classes C and C' as these are by definition bisimilar. The measure μ indicates how easy it is to replace class C by another one, or how good/precise is the attacker in distinguishing successor states. Then δ' is simply the weighted version of δ as described above with $\omega_{tC} = \mu(C)$. Note that there is no problem with the fact that δ' is defined recursively as we always know the δ' in the layers below before we compute δ' in the current layer.

Example 2. In order to illustrate how δ and δ' quantify the difference between various execution trees, let us consider the following four trees.



We abstract from the influence of different transition times and individual state labels, i.e. we assume that $t = 1$ for all transitions and that all states are labelled with the same label.

If we compute the δ and δ' values between all the pairs of systems we get the following results:

δ	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	$\mathbf{T}_4$
$\mathbf{T}_1$	0.000	0.500	1.000	0.000
$\mathbf{T}_2$	0.500	0.000	1.000	0.500
$\mathbf{T}_3$	1.000	1.000	0.000	1.000
$\mathbf{T}_4$	0.000	0.500	1.000	0.000

δ'	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	$\mathbf{T}_4$
$\mathbf{T}_1$	0.000	0.250	0.125	0.000
$\mathbf{T}_2$	0.250	0.000	0.125	0.250
$\mathbf{T}_3$	0.125	0.125	0.000	0.125
$\mathbf{T}_4$	0.000	0.250	0.125	0.000

From this we see that δ and δ' are symmetric, i.e. the difference between two systems is symmetric; that every system is bisimilar with itself, i.e. $\delta = 0 = \delta'$ (as we have an empty diagonal); and that the difference between two systems is between zero and one with values in between very well possible.

6 Cost Analysis

Our aim is to introduce "cost factors" into computer security. Instead of trying to achieve perfect security we will look at the trade-off between costs of security counter measures – such as increased average running time – and the improvement in terms of security, which we can measure via the δ or the weighted δ' introduced above.

6.1 Probabilistic Transformation

In [3] Agat introduces a program transformation to remove covert timing channels (*timing leaks*) from programs written in a sequential imperative programming language. He uses a language of security types with two security levels that is based on earlier work by Volpano and Smith [13,1]. Whilst Volpano and Smith restrict the condition in both while-loops and if-commands to being of the lowest security level, Agat allows the condition in an if-command to be high security providing that an external observer cannot detect which branch was taken. He shows that if a program is typeable in his system, then it is secure against timing attacks. This result depends critically on a notion of bisimulation; an if-command with a high security condition is only typeable if the two branches are bisimilar. Agat's notion of bisimilarity is timing aware and based on a notion of low-equivalence which ensures stepwise non-interference. He does not give an algorithm for bisimulation checking.

If a program fails to type, Agat presents a transformation system to remove the timing leak. The transformation pads the branches of if-commands with high security conditions with dummy commands. The objective of the padding is that both branches end up with the same timing and thus become indistinguishable by an external observer. The transformation utilises the concept of a *low-slice*: for a given command C, its low-slice C_L has the same syntactic structure as C but only has assignments to low security variables; all assignments to high security variables and branching on high security conditions are replaced by skip commands of appropriate duration. The transformation involves extending the branches in a high security if-command by adding the low-slice from the other branch. The effect of this transformation is that the timing of the execution of both branches are the same and equal to the sum of timing of the two branches in the untransformed program. Agat demonstrates that the transformation is semantically sound and that transformed programs are secure (correctness).

Rather than just adding the low slice from the other branch to each branch of a high security conditional, we transform each branch to make a probabilistic choice between its padded and untransformed variant. This allows us to trade-off the increased run-time of the padded program versus the vulnerability to attack of the untransformed program. The transformation described is just one on a whole spectrum of probabilistic transformations – at the other extreme we could probabilistically decide whether or not to execute each command in the low slice. All the formal transformation rules for probabilistic padding can be found in the full version [7]. The only rule which differs from the original semantics in [3] is the rule (If$_H$) given below. Here we replace – provided certain typing conditions are fulfilled – the branches of an **if** statement not just by the correctly "padded" version as in [3]; instead we introduce in every branch a choice such that the secure replacement will be executed only with probability p while with probability $1 - p$ the original code fragment will be executed.

$$\frac{\Gamma \vdash_{\leq} e : \mathbf{Bool}_H \quad \Gamma \vdash C_1 \hookrightarrow D_1 \mid D_{1L} \quad \Gamma \vdash C_2 \hookrightarrow D_2 \mid D_{2L} \quad ge(D_{1L}) = \emptyset \quad ge(D_{2L}) = \emptyset}{\Gamma \vdash \mathbf{if}\ (e)\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \hookrightarrow \begin{array}{l} \mathbf{if}\ (e)\ \mathbf{then}\ (\mathbf{choose}^p\ D_1\ \mathbf{or}\ D_1; D_{2L})\ \mathbf{else} \\ (\mathbf{choose}^p\ D_2\ \mathbf{or}\ D_{1L}; D_2) \mid \mathbf{skipIf}\ e\ (D_{1L}; D_{2L}) \end{array}}$$

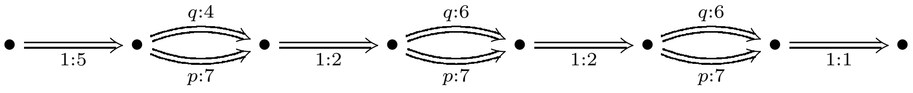
6.2 An Example

Our probabilistic version of Agat's padding algorithm allows us to obtain *partially* fixed programs. Depending on the parameter p with which we introduce empty low slices to obfuscate the timing leaks we can determine the (average) execution time of the fixed program in comparison with the improvement in security.

Agat presents in his paper [3] an example which itself is based on Kocher's study [2] of timing attacks against the RSA algorithm. In order to illustrate our approach we simplify the example slightly: The insecure program `agat` we start with is depicted on the left side in Table 2. The fully padded version Agat's algorithm produces, `fagat`, is on the right hand side of Table 2 (to keep things simple we omit Agat's empty statements like `skipAsn s s`; as `skip` as well as `s:=s` can be used just to 'spend time' without having any real effect on the store we can use e.g. `s:=s` in place of Agat's `skipAsn s s`). The program, `pagat`, presented in the middle of Table 2 is the result of *probabilistic padding*: The original program `agat` is transformed in such a way that the compensating statements, i.e. low slices, are executed only with probability p while with probability $q = 1 - p$ the original code is executed. For $p = 0$ we have the same behaviour as the original program `agat` while for $p = 1$ this program behaves in the same way as Agat's fully padded version `fagat`.

In our concrete experiments we used the following assumptions. The variable `i` can take values in $\{1, .., 4\}$ while `k` is a three dimensional array with values in $\{0, 1\}$ – nothing is concretely assumed about `s`. The variables `k`, representing a *secret key*, and `s` have security typing H, while `i` is the only low variable which can be observed by an attacker. We implemented this example using (arbitrary) execution times: $t_{asn} = 3$ (assign time), $t_{br} = 2$ (test/branch time), and $t_{skip} = 1$ (skip time), and $t_{ch} = 0$ (choice time).

The abstract semantics for the `pagat` program – which only records choice points and the moments in time when the low variable changes its value – produces the following execution trees if we start with keys `k=011` and `k=010`:



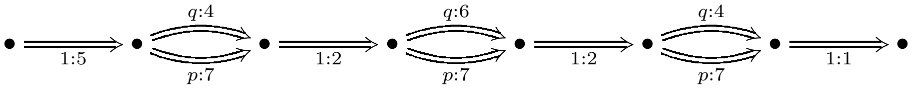


Table 2. Versions of Agat's Program: `agat`,`pagat`, and `fagat`

```
i := 1;
while i<=3 do
 if k[i]==1 then
   s := s;
 else
   skip;
 fi;
 i := i+1;
od;
```

```
i := 1;
while i<=3 do
 if k[i]==1 then
   choose p: s := s; skip
   or     q: s := s
   ro
 else
   choose p: skip
   or     q: s := s; skip
   ro
 fi;
 i := i+1;
od;
```

```
i := 1;
while i<=3 do
 if k[i]==1 then
   s := s; skip
 else
   s := s; skip
 fi;
 i := i+1;
od;
```

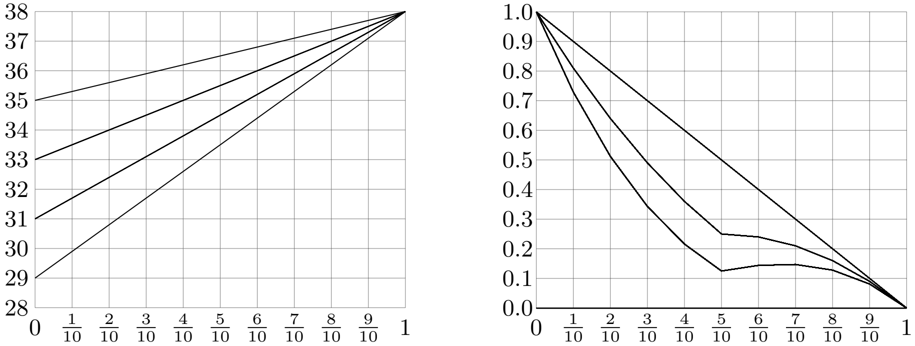


Fig. 1. Running Time $t(p)$ and Security Level $\delta'(p)$ as Functions of p

One can easily see from this how probabilistic padding influences the behaviour of a program: For every bit in the key `k` – i.e. every iteration – we have a choice between executing the original code with probability $q = 1 - p$ or the 'safe' code with probability p. The new code always takes the same time (in our case 7 ticks) while the original code's execution time depends on whether `k`[`i`] is set or not (either 4 or 6 time steps in our case). Clearly, for $p = 0$ we get in every iteration a different execution time, depending on the bit `k`[`i`], and thus can deduce the secrete value `k` by just observing the execution times. However, as the execution time is always the same for the replacement code, it is impossible to do the same for $p = 1$. For values of p between 0 and 1, the (average) execution times for `k`[`i`] $= 0$ and `k`[`i`] $= 1$ become more and more similar. This means in practical terms that the attacker has to spend more and more time (i.e. repeated observations of the program) in order to determine with high confidence the exact execution time and thus deduce the value of `k`[`i`] (cf. e.g. [9]).

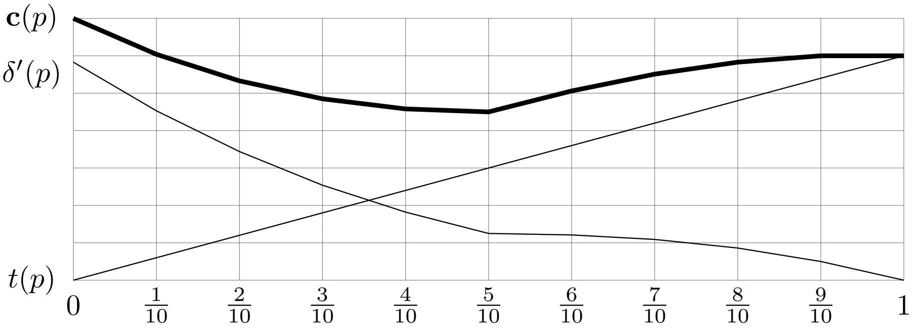
The price we have to pay for increased security, i.e. indistinguishability of behaviours, is an increased (average) execution time. The graph on the left in Figure 1 shows how the running time (vertical axis) increases in dependence of the padding probability p (horizontal axis) for the eight execution trees we have to consider in this example, i.e. for $\mathtt{k} = \mathtt{000}$, $\mathtt{k} = \mathtt{001}$, $\mathtt{k} = \mathtt{010}$, etc. Depending on the number of bits set in $\mathtt{k}$ we get four different curves which show how, for example for $\mathtt{k} = \mathtt{000}$ the running time increases from 29 time steps (for $p = 0$, i.e. `agat` program) to 38 (for $p = 1$, i.e. `fagat` program).

We can employ the bisimilarity measures δ and δ' in order to determine the security of the partially padded program. For this we compute using our algorithm $\delta(\mathtt{k}_i, \mathtt{k}_j)$ and $\delta'(\mathtt{k}_i, \mathtt{k}_j)$ for all possible keys, i.e. $i, j = 0, \ldots 7$. It turns out that $\delta = 1$ for all values of $p < 1$ and any pair of keys $\mathtt{k}_i$ and $\mathtt{k}_j$ with $i \neq i$; only for $p = 1$ we get, as one would expect, $\delta = 0$ for all key pairs. The weighted measure δ' is more sensitive. The $\delta'(\mathtt{k}_i, \mathtt{k}_i)$'s are, of course, all zero as every execution tree is bisimilar to itself. The other entries however are different from 0 and 1 and reflect the similarity between the two keys and thus the resulting execution trees. We get for example for $p = 0.5$ the following values for $\delta'(\mathtt{k}_i, \mathtt{k}_i)$:

δ'	000	001	010	011	100	101	110	111
000	0.000	0.125	0.250	0.125	0.500	0.125	0.250	0.125
001	0.125	0.000	0.125	0.250	0.125	0.500	0.125	0.250
010	0.250	0.125	0.000	0.125	0.250	0.125	0.500	0.125
011	0.125	0.250	0.125	0.000	0.125	0.250	0.125	0.500
100	0.500	0.125	0.250	0.125	0.000	0.125	0.250	0.125
101	0.125	0.500	0.125	0.250	0.125	0.000	0.125	0.250
110	0.250	0.125	0.500	0.125	0.250	0.125	0.000	0.125
111	0.125	0.250	0.125	0.500	0.125	0.250	0.125	0.000

If we plot the development of δ' as a function of p we observe only three patterns as depicted in the right graph in Figure 1. In all three cases δ' decreases from an original value 1 to 0, but in different ways.

In analysing the trade-off between increased running time and security we need to define a *cost* function. For example, one could be faced with a situation where a certain code fragment needs to be executed in a certain maximal time, i.e. there is a (cost) penalty if the execution takes longer than a certain number of micro-seconds. In our case we will consider a very trivial cost function $c(p) = 6\delta'(p) + t(p)$ with $\delta'(p)$ and $t(p)$ the average δ' between all possible execution trees and t the average running time. The following diagram depicts how $c(p)$, $\delta'(p)$ and $t(p)$ depend on the padding parameter p.



One can argue about the practical relevance of our particular cost function c. Nevertheless, this example illustrates already nicely the non-linear nature of security cost optimisation: The optimal, i.e. minimal, cost is reached in this case obviously for $p = 0.5$, i.e. keeping the cost of security counter measures in mind it is better to use a “half-fixed” program rather than a completely safe one.

7 Related and Further Work

The idea of defining a secure system via the requirement that an attacker must be unable to observe different behaviours as a result of different secrets – i.e. the system “operates in the same way” whatever value a secret key has – goes back at least to the work of Goguen and Meseguer [14].

This led in a number of settings to formalisations of security concepts such as “non-interference” via various notions of behavioural equivalencies (see e.g. [15,16]). One of the perhaps most prominent of these equivalence notions, namely *bisimilarity*, plays an important role in the context of security of concurrent systems but also found application for sequential programs such as in Agat’s work (as the interaction between system and attacker can be modelled as a parallel composition).

In order to allow for a decision theoretic analysis of security counter-measures and associated efforts it appears to be desirable to introduce a “quantitative” notion of the underlying behavioural equivalence. In the case of *bisimilarity* a first step was the introduction of the notion of *probabilistic bisimulation* by Larson and Skou [8]. However, this notion turns out to be still too strict and a number of researchers developed “approximate” versions; among them we just name the approaches by Desharnais et.al. [17,18] and van Breugel [19] and our work [10,20] (an extensive bibliography on this issue can be found in [21]). We based this current paper on the latter approach because it allows for an implementation of the semantics of pWhile via linear operators, i.e. matrices, and an efficient computation of δ and δ' using standard software such as `octave` [22].

Further research will be needed in order to clarify the relation between our measures δ and existing notions of *approximate bisimilarity* mentioned above, e.g. the ε in [9]. Furthermore, we also would like to shed more light on the relationship between our notion and information theoretic concepts used in the work of, for example Clark et.al. [23] and Boreale [24].

References

1. Smith, G., Volpano, D.: Secure information flow in a multi-threaded imperative language. In: POPL 1998, pp. 355–364 (1998)
2. Kocher, P.: 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)
3. Agat, J.: Transforming out timing leaks. In: POPL 2000, pp. 40–53 (2000)
4. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
5. Jonsson, B., Yi, W., Larsen, K.: Probabilistic extentions of process algebras. In: Handbook of Process Algebra, pp. 685–710. Elsevier Science, Amsterdam (2001)

6. Stirzaker, D.: Probability and Random Variables. Cambridge University Press, Cambridge (1999)
7. Di Pierro, A., Hankin, C., Wiklicky, H.: Quantifying timing leaks and cost optimisation. Technical Report arXiv:0807.3879 (2008)
8. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Information and Computation 94, 1–28 (1991)
9. Di Pierro, A., Hankin, C., Wiklicky, H.: Measuring the confinement of probabilistic systems. Theoretical Computer Science 340(1), 3–56 (2005)
10. Di Pierro, A., Hankin, C., Wiklicky, H.: Quantitative relations and approximate process equivalences. In: Amadio, R., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 508–522. Springer, Heidelberg (2003)
11. Paige, R., Tarjan, R.: Three partition refinement algorithms. SIAM Journal of Computation 16(6), 973–989 (1987)
12. Di Pierro, A., Hankin, C., Siveroni, I., Wiklicky, H.: Tempus fugit: How to plug it. Journal of Logic and Algebraic Programming 72(2), 173–190 (2007)
13. Volpano, D., Smith, G.: Confinement properties for programming languages. SIGACT News 29(3), 33–42 (1998)
14. Goguen, J., Meseguer, J.: Security Policies and Security Models. In: IEEE Symposium on Security and Privacy, pp. 11–20 (1982)
15. Ryan, P., Schneider, S.: Process algebra and non-interference. Journal of Computer Security 9(1/2), 75–103 (2001)
16. Focardi, R., Gorrieri, R.: Classification of Security Properties (Part I). In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 331–396. Springer, Heidelberg (2001)
17. Desharnais, J., Jagadeesan, R., Gupta, V., Panangaden, P.: Metrics for labeled markov systems. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 258–273. Springer, Heidelberg (1999)
18. Desharnais, J., Jagadeesan, R., Gupta, V., Panangaden, P.: The metric analogue of weak bisimulation for probabilistic processes. In: LICS 2002, pp. 413–422 (2002)
19. van Breugel, F.: A behavioural pseudometric for metric labelled transition systems. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 141–155. Springer, Heidelberg (2005)
20. Di Pierro, A., Hankin, C., Wiklicky, H.: Approximate Non-Interference. Journal of Computer Security 12(1), 37–81 (2004)
21. ABE 2008: Concur workshop on Approximate Behavioural Equivalences (2008), www.cse.yorku.ca/abe08
22. Eaton, J.W.: Octave. Technical report, Free Software Foundation, Boston, MA
23. Clark, D., Hunt, S., Malacaria, P.: Quantitative information flow, relations and polymorphic types. Journal of Logic and Computation 15(2), 181–199 (2005)
24. Boreale, M.: Quantifying information leakage in process calculi. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 119–131. Springer, Heidelberg (2006)

Method for Detecting Vulnerability to Doubling Attacks

Chong Hee Kim* and Jean-Jacques Quisquater

UCL Crypto Group, Université Catholique de Louvain, Belgium,
Place du Levant, 3, Louvain-la-Neuve, 1348, Belgium
{chong-hee.kim,jean-jacques.quisquater}@uclouvain.be

Abstract. The doubling attack by Fouque and Valette and its analogue, the relative doubling attack, by Yen et al. are a new kind of simple power analysis that can be applied to a binary double-and-add algorithm in a scalar multiplication (or a multiply-and-square algorithm in a modular exponentiation). The doubling attack is very powerful because it requires just two queries to the device to find the secret key. The original doubling attack broke the binary double-and-add always algorithm and the relative doubling attack succeeded in breaking the Montgomery ladder. Fouque and Valette told that the doubling attack was applicable only to downward algorithms, i.e., "left-to-right" implementations of a binary modular exponentiation and recommended to use upward "right-to-left" implementations. On the contrary, Yen et al. proposed a new downward algorithm and asserted that it was secure against doubling attacks. This kind of controversy comes from the lack of analysis of the fundamentals of the doubling attacks. Therefore we analyze the characteristic of the doubling attack and propose a method to easily test a given algorithm's security against doubling attacks. Furthermore, we show Yen et al.'s scheme is still vulnerable to the doubling attack.

Keywords: Doubling attack, relative doubling attack, modular exponentiation, simple power analysis (SPA), smart card.

1 Introduction

We can easily find cryptographic hardware devices such as smart cards everywhere in our daily lives from banking cards to SIM (Subscriber Identity Module) cards for GSM (Global System for Mobile communications). One of the main reasons why these cryptographic hardware devices are widely used is that they are believed to be tamper-resistant. This is why they host the implementation of many cryptographic protocols. However a recent development of physical attacks shows that a naive implementation of cryptographic protocols does not provide security anymore.

Side channel attacks using side channel information such as a computational timing and a power consumption of a device initiated by Kocher [12,11] are

[*] Supported by Walloon Region, Belgium / E.USER project.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 97–110, 2008.

now well studied. The simple power analysis (SPA) uses one or a few collected power consumption profiles of a device executing an algorithm and tries to find information on the secret key. A variant of SPA, called "doubling attack," was proposed by Fouque and Valette [6]. It targets an exponentiation algorithm or a scalar multiplication that is central in the implementation of public-key cryptography such as RSA, Diffie-Hellman, DSA, or ElGamal. The doubling attack is very powerful since it requires just two power consumption profiles with chosen messages to find the secret key. In this attack we assume that the adversary mounts a chosen ciphertext attack. This is a valid assumption in a side channel scenario, since a randomized padding avoiding chosen ciphertext attacks, such as OAEP [1,2], is checked after the running of the decryption process. As a consequence the exponentiation or multiplication is always performed and side channel information can be obtained.

The first victim to the doubling attack was the left-to-right square-and-multiply always algorithm [6]. Fouque and Valette said that the doubling attack only worked for "left-to-right", i.e., downward algorithm and recommended to use "right-to-left" algorithm combined with the appropriate countermeasures. Unfortunately they did not present the fundamental principle of the doubling attack, which caused a variant of the doubling attack by Yen et al. [15]. They proposed so called "relative doubling attack" and showed that the Montgomery ladder algorithm [10] was vulnerable. Furthermore they proposed a new left-to-right exponentiation algorithm and asserted it was secure against the doubling attacks even though it is a left-to-right implementation.

This kind of controversy comes from a lack in the study on the fundamental principle of the doubling attack. Therefore we analyze the characteristic of the doubling attack and propose a method to easily check a given algorithm's security against doubling attacks. With this method we show why the left-to-right always algorithm and the Montgomery ladder algorithm are vulnerable and the right-to-left algorithm is secure. Furthermore we show that Yen et al.'s proposed algorithm is also vulnerable to the doubling attack. Therefore our method can be a good measure to check the security against doubling attacks when someone devises a new exponentiation algorithm.

2 Exponentiation Algorithms and Doubling Attack

A modular exponentiation is the main operation in RSA or the discrete logarithm problem. In the elliptic curve setting, the corresponding operation is a scalar multiplication. From an algorithmic point of view, those two operations are very similar; the only difference is the underlying group structure. We consider the problem of computing a modular exponentiation in this paper. However all what we state can be easily transposed to a scalar multiplication.

2.1 Exponentiation Algorithms and Simple Power Analysis

Although numerous exponentiation algorithms have been developed (see [9] for a survey), practical solutions for devices with constrained computation and storage capabilities (e.g., smart card) are usually restricted to the *square and multiply algorithms* (double and add algorithm in a point scalar multiplication). Let $\sum_{i=0}^{n-1} d_i \cdot 2^i$ with $d_i \in \{0,1\}$ be the binary expansion of exponent d. There are two versions of square and multiply algorithms.

The left-to-right (MSB-to-LSB) binary algorithm as shown in Fig. 1 is especially preferable to implementations in smart cards because this algorithm needs only one temporary memory $S[0]$ if the input data M is stored inside the smart cards. Contrary to the left-to-right binary algorithm, the right-to-left binary algorithm starts the computation from the least significant bit of the exponent as shown in Fig. 2. Furthermore it requires two temporary memories $S[0]$ and $S[1]$.

```
INPUT: $M \neq 0, d = (d_{n-1}, ..., d_0)_2, N$
OUTPUT: $M^d$ mod $N$

$S[0] \leftarrow 1$
for $i$ from $n-1$ to $0$ do
    $S[0] \leftarrow S[0]^2$ mod $N$
    if $d_i = 1$ then
        $S[0] \leftarrow S[0] \cdot M$ mod $N$
return $S[0]$
```

Fig. 1. Left-to-right binary algorithm

```
INPUT: $M \neq 0, d = (d_{n-1}, ..., d_0)_2, N$
OUTPUT: $M^d$ mod $N$

$S[0] \leftarrow 1$, $S[1] \leftarrow M$
for $i$ from $0$ to $n-1$ do
    if $d_i = 1$ then
        $S[0] \leftarrow S[0] \cdot S[1]$ mod $N$
    $S[1] \leftarrow S[1]^2$ mod $N$
return $S[0]$
```

Fig. 2. Right-to-left binary algorithm

Simple power analysis (SPA) exploits distinct patterns from a single power trace [12]. The basic left-to-right binary algorithm of Fig. 1 and right-to-left binary algorithm of Fig. 2 are vulnerable to SPA. Because depending on the value of a bit d_i, a multiplication operation is performed. Therefore the attacker can easily compute the value of the secret key with the observation of one power consumption profile. By balancing the operations regardless of the value of the secret bit d_i you can prevent SPA. A simple example to prevent SPA is shown in Fig. 3.

2.2 Doubling Attack

Fouque and Valette presented a new kind of power analysis attack, so called *doubling attack*, on the "left-to-right" exponentiation [6]. By querying two chosen messages, the attacker can recover all the secret data. The idea of the attack is based on the fact that, even if an adversary is not able to tell which computation is done by the card, he can at least detect when the card does twice the same operation. More precisely, if the card computes A^2 and B^2, the attacker is not able to guess the value of A nor B but he is able to check if $A = B$.

Let us consider the left-to-right algorithm described in Fig. 3. Let us denote the partial sums $S[0]_k(M)$ as the intermediate value stored in $S[0]$ after $k+1$ iterations when the input is M. Then we have

```
INPUT: $M \neq 0, d = (d_{n-1}, ..., d_0)_2, N$
OUTPUT: $M^d$ mod $N$

$S[0] \leftarrow 1$
for $i$ from $n-1$ to 0 do
    $S[0] \leftarrow S[0]^2$ mod $N$
    $S[\overline{d_i}] \leftarrow S[\overline{d_i}] \cdot M$ mod $N$
return $S[0]$
```

Fig. 3. Left-to-right square-and-multiply always algorithm

```
INPUT: $M \neq 0, d = (d_{n-1}, ..., d_0)_2, N$
OUTPUT: $M^d$ mod $N$

$S[0] \leftarrow 1$, $S[1] \leftarrow M$
for $i$ from $n-1$ to 0 do
    $S[\overline{d_i}] \leftarrow S[\overline{d_i}] \cdot S[d_i]$ mod $N$
    $S[d_i] \leftarrow S[d_i]^2$ mod $N$
return $S[0]$
```

Fig. 4. Montgomery ladder algorithm

$$\begin{aligned} S[0]_k(M) &= (M)^{\sum_{i=0}^{k} d_{n-1-i} 2^{k-i}} \\ &= (M^2)^{\sum_{i=0}^{k-1} d_{n-1-i} 2^{k-1-i}} \cdot (M)^{d_{n-1-k}} \\ &= S[0]_{k-1}(M^2) \cdot (M)^{d_{n-1-k}}. \end{aligned}$$

The intermediate result with M at step k will be equal to the intermediate result with M^2 at step $k-1$ if and only if d_{n-1-k} is null. We just need to compare the doubling computation at step $k+1$ for M and at step k for M^2 to recover the bit d_{n-1-k}. If both computations are identical, d_{n-1-k} is equal to 0 otherwise d_{n-1-k} is equal to 1. Therefore, with only two requests to the card, it is possible to recover all the bits of the secret exponent.

Let us consider an example. Let $d = 78 = 1001110_{(2)}$ and focus on the doubling operations of Fig. 3. As shown in Table 1, when $d_i = 0$ the doubling operation for M^2 at step i and that for M at step $i+1$ are the same.

The doubling attack enables recovering the secret key decryption of RSA [14], the key decryption of ElGamal [5], or the secret key of the Diffie-Hellman authentication system. It only focuses on the decryption cases. Since the randomized padding avoiding chosen ciphertext attacks, such as OAEP [1,2], is checked after the running of the decryption process. Therefore the attacker can still collect the necessary power traces.

2.3 Relative Doubling Attack

In 2002, Joye and Yen published a safe error and SPA resistant exponentiation algorithm [10]. As shown in Fig. 4, it does not have any redundant operations. If a fault is induced at any moment of the exponentiation then the result will always be corrupted. Thus, it is impossible to obtain a safe error. Furthermore it performs a multiplication and a squaring regardless of the value of the secret bit, which prevents SPA.

In 2005, Yen et al. showed that the Montgomery ladder algorithm was vulnerable to a variant of the doubling attack, so called the *relative doubling attack* [15]. The assumption is basically the same as what is considered in the

Table 1. Example of doubling attack

i	d_i	M^d	$(M^2)^d$
6	1	$S[0] \leftarrow (1)^2$ $S[0] \leftarrow 1 \times M$	$S[0] \leftarrow (1)^2$ $S[0] \leftarrow 1 \times M^2$
5	0	$S[0] \leftarrow (M)^2$ $S[1] \leftarrow M^2 \times M$	$S[0] \leftarrow (\mathbf{M}^{\mathbf{2}})^{\mathbf{2}}$ $S[1] \leftarrow M^4 \times M^2$
4	0	$S[0] \leftarrow (\mathbf{M}^{\mathbf{2}})^{\mathbf{2}}$ $S[1] \leftarrow M^4 \times M$	$S[0] \leftarrow (\mathbf{M}^{\mathbf{4}})^{\mathbf{2}}$ $S[1] \leftarrow M^8 \times M^2$
3	1	$S[0] \leftarrow (\mathbf{M}^{\mathbf{4}})^{\mathbf{2}}$ $S[0] \leftarrow M^8 \times M$	$S[0] \leftarrow (M^8)^2$ $S[0] \leftarrow M^{16} \times M^2$
2	1	$S[0] \leftarrow (M^9)^2$ $S[0] \leftarrow M^{18} \times M$	$S[0] \leftarrow (M^{18})^2$ $S[0] \leftarrow M^{36} \times M^2$
1	1	$S[0] \leftarrow (M^{19})^2$ $S[0] \leftarrow M^{38} \times M$	$S[0] \leftarrow (M^{38})^2$ $S[0] \leftarrow M^{76} \times M^2$
0	0	$S[0] \leftarrow (M^{39})^2$ $S[1] \leftarrow M^{78} \times M$	$S[0] \leftarrow (M^{78})^2$ $S[1] \leftarrow M^{156} \times M^2$
Return		M^{78}	M^{156}

doubling attack [6]; an adversary can distinguish collision of power trace segments (within a single or more power traces) when the smart card performs twice the same computation even if the adversary is not able to tell exactly which computation is done. An adversary is assumed to be able to detect the collision of $A^2 \bmod N$ and $B^2 \bmod N$ if $A = B$ even though A and B are unknown.

The relative doubling attack uses an approach to derive the private key in which the relationship between two adjacent private key bits can be obtained as either $d_i = d_{i-1}$ or $d_i \neq d_{i-1}$. An example of assuming the private exponent d to be $75 = (1, 0, 0, 1, 0, 1, 1)_2$ and two related input data to be M and M^2 respectively, is shown in Table 2. The computational process of raising M^d and $(M^2)^d$ reveals the fact that given $d_0 = 1$ and the observation of collision of the iteration d_0 of M^d and iteration d_1 of $(M^2)^d$ will lead to the result of $d_1 = d_0 = 1$.

The original doubling attack (against the square-and-multiply always algorithm) focuses on deriving the private key bit d_i by checking whether $d_i = 0$. So, the original doubling attack tries to obtain the knowledge of *absolute* value of each d_i. On the contrary, the relative doubling attack (against the Montgomery ladder algorithm) focuses on deriving the knowledge of whether $d_i = d_{i-1}$ (relationship between every two adjacent key bits), but not the knowledge of either d_i or d_{i-1} directly.

Table 2. Example of relative doubling attack

i	d_i	M^d	$(M^2)^d$
6	1	$S[0] \leftarrow 1 \times M$ $S[1] \leftarrow (M)^2$	$S[0] \leftarrow 1 \times M^2$ $S[1] \leftarrow (M^2)^2$
5	0	$S[1] \leftarrow M^2 \times M$ $S[0] \leftarrow (M)^2$	$S[1] \leftarrow M^4 \times M^2$ $S[0] \leftarrow (\mathbf{M^2})^\mathbf{2}$
4	0	$S[1] \leftarrow M^3 \times M^2$ $S[0] \leftarrow (\mathbf{M^2})^\mathbf{2}$	$S[1] \leftarrow M^6 \times M^4$ $S[0] \leftarrow (M^4)^2$
3	1	$S[0] \leftarrow M^4 \times M^5$ $S[1] \leftarrow (M^5)^2$	$S[0] \leftarrow M^8 \times M^{10}$ $S[1] \leftarrow (M^{10})^2$
2	0	$S[1] \leftarrow M^{10} \times M^9$ $S[0] \leftarrow (M^9)^2$	$S[1] \leftarrow M^{20} \times M^{18}$ $S[0] \leftarrow (M^{18})^2$
1	1	$S[0] \leftarrow M^{18} \times M^{19}$ $S[1] \leftarrow (M^{19})^2$	$S[0] \leftarrow M^{36} \times M^{38}$ $S[1] \leftarrow (\mathbf{M^{38}})^\mathbf{2}$
0	1	$S[0] \leftarrow M^{37} \times M^{38}$ $S[1] \leftarrow (\mathbf{M^{38}})^\mathbf{2}$	$S[0] \leftarrow M^{74} \times M^{76}$ $S[1] \leftarrow (M^{76})^2$
Return		M^{75}	M^{150}

3 Fundamentals of Doubling Attack

Although the doubling attack and the relative doubling attack succeeded in breaking two exponentiation algorithms, there is no fundamental principle or detail analysis of the doubling attacks until now. Even the creators of the doubling attack, Fouque and Valette, did not tell anything about the fundamentals of doubling attack. Therefore some people just assume that downward algorithms may be vulnerable to doubling attacks without certainty. Others assert that their downward scheme is secure against doubling attacks without a proof [15].

In this section, we analyze the fundamentals of the doubling attack and show how we can easily test a given algorithm's security against the doubling attacks.

3.1 Characteristics of Doubling Attack

Doubling attacks require two requirements. Their fundamental principle relies on the consecutive squaring (doubling in a point scalar multiplication) operations. If there are two consecutive squaring operations related to the value of the secret key (an upward algorithm is secure against doubling attacks because a squaring operation is done regardless of the value of the secret key, we will discuss it later), we can apply doubling attacks. In addition, the intermediate value of computing $(M^2)^d$ should be the exactly square of (double in a point scalar multiplication) that of computing $(M)^d$ after each iteration.

Let us denote $S_k(M)$ and $S_k(M^2)$ as the k^{th} intermediate values of computing $(M)^d$ mod N and $(M^2)^d$ mod N respectively. And we suppose that $S_k(M^2) = (S_k(M))^2$. Then if there are two consecutive squaring operations, we have:

$$S_{k+1}(M) = (S_k(M))^2 \bmod N, \tag{1}$$
$$S_{k+2}(M) = (S_{k+1}(M))^2 \bmod N, \tag{2}$$

and

$$S_{k+1}(M^2) = (S_k(M^2))^2 \bmod N, \tag{3}$$
$$S_{k+2}(M^2) = (S_{k+1}(M^2))^2 \bmod N. \tag{4}$$

Since $S_{k+1}(M) = (S_k(M))^2 = S_k(M^2)$, equation (2) and equation (3) are the same. Therefore, we can find the moment when equation (2) and equation (3) are performed in a device by comparing two power consumption profiles. If the occurrence of these two consecutive squaring operations depends on the value of the secret key, we can get information on the secret key through the doubling attacks.

Consequently, we can summarize the conditions for the success of the doubling attacks as follows:

1. two consecutive squaring operations (doubling in a point scalar multiplication) related to secret key bits should exist,
2. the intermediate value of computing $(M^2)^d$ should be exactly the square of (double in a point scalar multiplication) that of computing $(M)^d$ after each iteration.

For example, let us consider the left-to-right square-and-multiply always algorithm as shown in Fig. 3. According to the value of the bit d_i, the operations of each iteration are different. We can make the characteristic table of it as shown in Table 3, where S stands for "squaring" and M stands for "multiplication."

Table 3. Characteristic table of square-and-multiply always algorithm

		S[0]	
d_i	d_{i-1}	Operation at d_i	Operation at d_{i-1}
0	0	**S**	**S**
0	1	**S**	**S**, M
1	0	S, M	S
1	1	S, M	S, M

Although two variables $S[0]$ and $S[1]$ are used, only $S[0]$ satisfy the second condition; intermediate value of computing $(M^2)^d$ mod N is the square of that of computing $(M)^d$ mod N at each iteration. To prove this, let us denote the partial sums $S[0]_k(M)$ as the intermediate value stored in $S[0]$ after k iterations when the input is M. Then we have

$$S[0]_k(M) = (M)^{\sum_{i=0}^{k} d_{n-1-i}2^{k-i}}.$$

Similarly, we have partial sums $S[0]_k(M^2)$ as the intermediate value stored in $S[0]$ after k iterations when the input is (M^2) as follows;

$$\begin{aligned} S[0]_k(M^2) &= (M^2)^{\sum_{i=0}^{k} d_{n-1-i}2^{k-i}} \\ &= ((M)^{\sum_{i=0}^{k} d_{n-1-i}2^{k-i}})^2 \\ &= (S[0]_k(M))^2. \end{aligned}$$

Therefore we can say that the left-to-right square-and-multiply always algorithm satisfies the condition (2). Next we consider the condition (1). We can find two cases, $(d_i, d_{i-1}) = (0, 0)$ and $(d_i, d_{i-1}) = (0, 1)$, of the two consecutive squaring operations in Table 3. More simply, we can say that whenever d_i equals 0 we have two consecutive squaring operations. Therefore we can conclude that the left-to-right square-and-multiply always algorithm is vulnerable to the doubling attack.

For the second example, we make the characteristic table of the Montgomery ladder as shown in Table 4. Both intermediate values $S[0]$ and $S[1]$ satisfy the condition (2): intermediate value of computing $(M^2)^d$ mod N is the square of that of computing $(M)^d$ mod N at each iteration. The two consecutive squaring operations are shown in $S[0]$ when $d_i = d_{i-1} = 0$ and in $S[1]$ when $d_i = d_{i-1} = 1$. Therefore the Montgomery ladder is vulnerable to the doubling attack.

Table 4. Characteristic table of Montgomery ladder

		S[0]		S[1]	
d_i	d_{i-1}	Operation at d_i	Operation at d_{i-1}	Operation at d_i	Operation at d_{i-1}
0	0	**S**	**S**	M	M
0	1	S	M	M	S
1	0	M	S	S	M
1	1	M	M	**S**	**S**

3.2 Upward vs. Downward Algorithm

We compare upward and downward exponentiation algorithms with respect to the requirements of the doubling attacks we proposed in the previous section.

The downward computation of M^d is based on the following structure:

$$M^d = ((((M^{d_{n-1}})^2) \cdot M^{d_{n-2}})^2 \cdots M^{d_1})^2 \cdot M^{d_0}. \tag{5}$$

From d_{n-1} to d_0, it multiplies the intermediate value by M^{d_i} and then squares the intermediate value. Therefore only if d_i equals 0, then two consecutive squaring operations occur. Furthermore the intermediate value of computing $(M^2)^d$

is the square of the intermediate value of computing M^d. Therefore a downward computation is vulnerable to the doubling attacks.

Montgomery ladder is also based on the equation (5). It computes both a squaring and a multiplication at each iteration. The allocation of each result is decided by the current bit of the secret key. Therefore two consecutive squaring operations happen whenever two consecutive bits of the secret key are the same, which makes in turn Montgomery ladder vulnerable to the doubling attacks.

The upward exponentiation is based on the following structure:

$$M^d = (M)^{d_0} \cdot (M^2)^{d_1} \cdot (M^{2^2})^{d_2} \cdots (M^{2^{n-1}})^{d_{n-1}}. \tag{6}$$

There are two variables. One variable, $S[0]$, stores the intermediate result and the other variable, $S[1]$, stores M^{2^i}. From d_0 to d_{n-1}, $S[1]$ computes M^{2^i} and $S[0]$ is multiplied by $S[1]$ depending on the value of d_i. We can easily see that two consecutive squaring operations occur at each time in $S[1]$ as shown in Fig. 2. However this does not depend on the secret key bits. The consecutive two squaring operations are performed at each iteration regardless of the secret key. Therefore it does not give any information on the secret key.

3.3 Remarks on Possible Countermeasures

The doubling attack requires at least two queries with known messages to the device. Therefore the best way to avoid it is to remove the relation between known messages and power consumption profiles by the randomization techniques. One randomization technique is to blind the secret exponent, for example, like $d' = d + r \cdot \phi(N)$ in RSA exponentiation. Similarly Coron's countermeasure, $d' = d + r \cdot \mathbb{E}$, uses a 20-bit random value r in a scalar multiplication [4]. However it is clearly pointed out in [6] that although Coron's countermeasure was sufficient to resist against usual DPA attacks, it was possible to identify right pairs with 2^{20} requests with the doubling attack. Therefore the security against doubling attacks relies on the size of a random number used to blind the secret key.

The second countermeasure is to blind the base, i.e., blinding of the point (the computation of $d(P + R)$ followed by a subtraction of dR, where R is a random point) in a scalar multiplication and blinding of the message (the computation $(Mr^e)^d$ followed by a multiplication of r^{-1}, where r is a random integer) in an exponentiation. Applying this method in a scalar multiplication requires twice the time needed for a single scalar multiplication. In an exponentiation, it requires an expensive inverse operation. Therefore to be more efficient, storing and updating random data are often used. For example, it is proposed to store a random point R and the associated value $S' = dR$ and update them as $R \leftarrow (-1)^b.2.R$ and $S' \leftarrow (-1)^b.2.S'$, where $b \in \{0, 1\}$, each time when they are used [4]. However it was shown to be vulnerable to the doubling attack in [6]. Therefore, to resist the doubling attacks a random update should be used as $R \leftarrow (-1)^b.a.R$ and $S' \leftarrow (-1)^b.a.S'$, where a is a random integer. Again, the security relies on the size of a.

The third countermeasure is to blind the modulus in an exponentiation. Giraud presented the use of $M^d \bmod k \cdot N$, where k is a 32-bit random number [8]. All modular multiplications are done with $k \cdot N$ except the final one that is done with N.

Consequently we know that the randomization techniques require a sufficient size of random data to resist the doubling attacks. However this degrades the efficiency of a device. If we use an exponent blinding, an extra computation time is required. The message blinding needs an extra computation and storage. The modulus blinding requires bigger operands as well as an extra computation time. The use of bigger operands may be a problem when a hardware multiplier is implemented. That is, we have to construct a hardware multiplier of bigger operands such as a 1056 bit or higher precision multiplier for a 1024 bit multiplication.

4 Doubling Attack on Yen et al.'s Algorithm

Yen et al. said in [15] that upward exponentiation was not a necessary requirement meant to be immune from the doubling attacks and showed a new downward SPA-protected and safe-error-protected exponentiation algorithm. They also asserted that it was secure against doubling and the relative doubling attacks. Unfortunately, there was no detailed proof of security against doubling attacks. In this section, we show that their algorithm is vulnerable to the doubling attack. We used the method we developed in the previous section to analyze its security.

4.1 Yen et al.'s Algorithm

Yen et al.'s algorithm [13,15,16] is a *balanced* downward "left-to-right" exponentiation algorithm. Since its structure of operations as shown in Fig. 5 is well balanced, it is immune to SPA. Furthermore it does not have any redundant operations that cause the safe-error-attack. One outstanding feature of Yen et al.'s algorithm is that the current operations of i^{th} iteration depend on the current bit d_i and the next bit d_{i-1}. There are two operations per iteration and the first operation relies on the value of d_i and the second operation relies on that of d_{i-1}.

```
INPUT: $M \neq 0, d = (d_{n-1}, ..., d_0)_2, d_{n-1} = 1, N$
OUTPUT: $M^d \bmod N$

$S[0] \leftarrow 1$, $S[1] \leftarrow M$, $d_{-1} \leftarrow 1$
for $i$ from $n-1$ to 0 do
    $S[0] \leftarrow S[0] \cdot S[\overline{d_i}] \bmod N$
    $S[0] \leftarrow S[0] \cdot S[d_{i-1}] \bmod N$
return $S[0]$
```

Fig. 5. Yen et al.'s algorithm

4.2 Proposed Attack

To check the security against doubling attacks, we first check the condition (2). We can easily see that after each iteration, the value stored in $S[0]$ of computing $(M^2)^d$ is the square of that of computing $(M)^d$. Therefore we can construct the characteristic table of Yen et al.'s algorithm as shown in Table 5. Three consecutive bits d_i, d_{i-1}, and d_{i-2} are used to construct the characteristic table since the operations at i^{th} iteration depend on (d_i, d_{i-1}) and the operations at $(i+1)^{th}$ iteration depend on (d_{i-1}, d_{i-2}). From the characteristic table we can see that there are two consecutive squaring operations when two consecutive bits are $(1, 0)$. Therefore we can find two bits $(1, 0)$ by comparing two power profiles of computing M^d and $(M^2)^d$.

Table 5. Characteristic table of Yen et al.'s algorithm

			S[0]	
d_i	d_{i-1}	d_{i-2}	Operation at d_i	Operation at d_{i-1}
0	0	0	M, S	M, S
0	0	1	M, S	M, M
0	1	0	M, M	**S, S**
0	1	1	M, M	S, M
1	0	0	**S, S**	M, S
1	0	1	**S, S**	M, M
1	1	0	S, M	**S, S**
1	1	1	S, M	S, M

An example of an attack assuming the private exponent d to be $107 = (1, 1, 0, 1, 0, 1, 1)_2$ and two related input data to be M and M^2 respectively, is shown in Table 6. At $i = 5$, we can find the same squaring operation of $(M^2)^2$ in both computing M^d and $(M^2)^d$. Therefore we can conclude $d_5 = 1$ and $d_4 = 0$. Similarly we can find d_3 and d_2. By assuming the most significant bit and the least significant equal 1, we remain d_1 unknown which can be recovered by an exhaustive search.

We can improve further than an exhaustive search to find the remaining bits. We know one case when two consecutive bits is (1,0), which is out of four cases, (0,0), (0,1), (1,0), and (1,1). Therefore, if the number of secret bits is n, then we know $n/4$ bits in average. Furthermore we can reduce the number of unknown bits. Since we can find two consecutive bits of (1,0), we can reduce the candidates for the bits between two (1,0)'s. For example, we suppose that we have found $(1, 0, X, X, X, 1, 0, X, X, 1, 0, X, X, X, X, 1, 0, 1)_2$ by doubling attack. Where, X means an unknown key bit, i.e., $X \in \{0, 1\}$. Since we can find all the (1,0)'s, the stream $(X, ..., X)$ does not contain any (1,0). The candidates for the stream $(X, ..., X)$ are as follows:

$$\begin{aligned}(X, X) &= (0, 0), (0, 1), (1, 1)\\ (X, X, X) &= (0, 0, 0), (0, 0, 1), (0, 1, 1), (1, 1, 1)\\ (X, X, X, X) &= (0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 1), (0, 1, 1, 1), (1, 1, 1, 1)\end{aligned}$$

Therefore we have $k + 1$ candidates for each unknown consecutive k bits. Although the number of unknown key bits is 10 in this example, we need to try $(3 \cdot 4 \cdot 5)$ candidates instead of 2^{10}.

Consequently, if (1,0) is uniformly distributed among n bits, then the number of bits between two (1,0)'s is 6. And there are $n/8$ sequences of 6 unknown consecutive bits in average. Therefore $(6 + 1)^{n/8}$ candidates remain in average, which equals $2^{0.35n}$ operations. Therefore, we know $n/4$ bits and require $2^{0.35n}$ operations to find the remaining bits in average. This partial key information can be used to find the whole key such as the attack of RSA with a partial key as shown in [3].

Table 6. Example of proposed attack on Yen et al.'s algorithm

i	d_i	M^d	$(M^2)^d$
6	1	$S[0] \leftarrow (1)^2$ $S[0] \leftarrow 1 \times M$	$S[0] \leftarrow (1)^2$ $S[0] \leftarrow 1 \times (M^2)$
5	1	$S[0] \leftarrow (M)^2$ $S[0] \leftarrow (\mathbf{M^2})^{\mathbf{2}}$	$S[0] \leftarrow (\mathbf{M^2})^{\mathbf{2}}$ $S[0] \leftarrow (M^4)^2$
4	0	$S[0] \leftarrow M^4 \times M$ $S[0] \leftarrow M^5 \times M$	$S[0] \leftarrow M^8 \times M^2$ $S[0] \leftarrow M^{10} \times M^2$
3	1	$S[0] \leftarrow (M^6)^2$ $S[0] \leftarrow (\mathbf{M^{12}})^{\mathbf{2}}$	$S[0] \leftarrow (\mathbf{M^{12}})^{\mathbf{2}}$ $S[0] \leftarrow (M^{24})^2$
2	0	$S[0] \leftarrow M^{24} \times M$ $S[0] \leftarrow M^{25} \times M$	$S[0] \leftarrow M^{48} \times M^2$ $S[0] \leftarrow M^{50} \times M^2$
1	1	$S[0] \leftarrow (M^{26})^2$ $S[0] \leftarrow M^{52} \times M$	$S[0] \leftarrow (M^{52})^2$ $S[0] \leftarrow M^{104} \times M^2$
0	1	$S[0] \leftarrow (M^{53})^2$ $S[0] \leftarrow M^{106} \times M$	$S[0] \leftarrow (M^{106})^2$ $S[0] \leftarrow M^{214} \times M^2$
Return		M^{107}	M^{214}

5 Conclusions

The doubling attack is a very powerful attack since it requires just two power consumption profiles with chosen messages. They can even avoid a randomized padding scheme such as OAEP [1,2]. Nonetheless there was no fundamental research on its principle. It has just been assumed that it works only on downward algorithms [6]. However Yen et al. brought forward a counterargument and proposed a new downward algorithm asserted to be secure against the doubling attacks [15].

In this paper, we analyzed the characteristic of the doubling attack and showed the fundamental principle of it, which brought a method to check a given algorithm's security against the doubling attacks. This method is very simple and easy to use. Therefore it can be a good measure to check the security against doubling attacks when someone devises a new algorithm. Finally we have shown with our method that Yen et al.'s algorithm was vulnerable to the doubling attack.

References

1. PKCS # 1, v2.1, RSA Cryptogrpaphy Standards (January 5, 2001), `http://www.rsasecurity.com/rsalabs/pkcs/`
2. Bellare, M., Rogaway, P.: Optimal asymmetric encrption padding - How to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
3. Blömer, J., May, A.: New partial key exposure attacks on RSA. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 27–43. Springer, Heidelberg (2003)
4. Coron, J.: Resistance against differential power analysis for elliptic curve. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
5. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logatihms. IEEE Transactions on Information Theory (4), 469–472 (1985)
6. Fouque, P.-A., Valette, F.: The doubling attack - why upwards is better than downwards. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 269–280. Springer, Heidelberg (2003)
7. Giraud, C.: Fault resistant RSA implementation. In: Breveglieri, L., Koren, I. (eds.) Second Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC 2005, pp. 142–151 (2005)
8. Giraud, C.: An RSA implementation resistant to fault attacks and to simple power analysis. IEEE Transactions on computers 55(9), 1116–1120 (2006); An earlier version appears in [7]
9. Gordon, D.: A survey of fast exponentiation methods. Journal of Algorithms 27, 129–146 (1998)
10. Joye, M., Yen, S.-M.: The montgomery powering ladder. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2003)
11. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSA and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

13. Lu, C.-C., Tseng, S.-Y., Huang, S.-K.: A secure modular exponential algorithm resists to power, timing, C safe error and M safe error attacks. In: 19^{th} International Conference on Advanced Information Networking and Applications (AINA 2005), vol. 2, pp. 151–154 (2005)
14. Rivest, A.S.R.L., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
15. Yen, S.-M., Ko, L.-C., Moon, S., Ha, J.: Relative doubling attack against Montgomery Ladder. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 117–128. Springer, Heidelberg (2006)
16. Yen, S.-M., Lu, C.-C., Tseng, S.-Y.: Method for protecting public key schemes from timing, power, and fault attacks. U.S. Patent Number US2004/0125950 A1 (July 2004)

Side Channel Analysis of Some Hash Based MACs: A Response to SHA-3 Requirements

Praveen Gauravaram[1,⋆] and Katsuyuki Okeya[2]

[1] DTU Mathematics, Technical University of Denmark, Denmark
p.gauravaram@mat.dtu.dk
[2] Hitachi, Ltd., Systems Development Laboratory, Japan
katsuyuki.okeya.ue@hitachi.com

Abstract. The forthcoming NIST's Advanced Hash Standard (AHS) competition to select SHA-3 hash function requires that each candidate hash function submission must have at least one construction to support FIPS 198 HMAC application. As part of its evaluation, NIST is aiming to select either a candidate hash function which is more resistant to known side channel attacks (SCA) when plugged into HMAC, or that has an alternative MAC mode which is more resistant to known SCA than the other submitted alternatives. In response to this, we perform differential power analysis (DPA) on the possible smart card implementations of some of the recently proposed MAC alternatives to NMAC (a fully analyzed variant of HMAC) and HMAC algorithms and NMAC/HMAC versions of some recently proposed hash and compression function modes. We show that the recently proposed BNMAC and KMDP MAC schemes are even weaker than NMAC/HMAC against the DPA attacks, whereas multi-lane NMAC, EMD MAC and the keyed wide-pipe hash have similar security to NMAC against the DPA attacks. Our DPA attacks do not work on the NMAC setting of MDC-2, Grindahl and MAME compression functions.

Keywords: Applied cryptography, hash functions, side channel attacks, HMAC.

1 Introduction

The cryptanalysis of the MD5 and SHA-1 hash functions [41,42] and its impact on several applications [5,10,13,16,40] have triggered a kind of feeding frenzy among the cryptographers. On the other hand, generic attacks [20,23] on the popular Merkle-Damgård (MD) hash framework [12,30] have exposed several of its undesirable properties.

In the wake of this active cryptanalysis of hash functions and its impact on applications, NIST is conducting an international competition to define an Advanced Hash Standard (AHS) which would be referred to as SHA-3 family [34].

[⋆] Author and this research project are supported by The Danish Research Council for Technology and Innovation grant number 274-08-0052.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 111–127, 2008.

NIST requires that each candidate hash function must have at least one construction to support the current applications of hash functions specified in the FIPS or NIST special publications that include FIPS 198 HMAC [33]. As part of its evaluation, NIST is also considering side channel attacks (SCA) on the hash based MACs. NIST intends to select as SHA-3, either a candidate hash which is more resistant to known SCA attacks when plugged into HMAC, or that has an alternative MAC mode which is more resistant to known SCA attacks than the other submitted alternatives [9,21,22].

Considering this state of the art of research in hash functions, we believe that AHS competition would receive hash as well as compression function modes as candidates for SHA-3 using structures and chaining modes different from the ones used in the broken hash functions. It is prominent that such proposals define provably secure MAC modes with a protection from the SCA attacks.

This research problem has motivated us to assess the security of several recently proposed MAC alternatives to NMAC (a thoroughly analysed variant of HMAC) and HMAC algorithms [3,2] and some compression function and hash function modes in the NMAC/HMAC setting from differential power analysis (DPA) attacks. We analyse MACs that are assumed to be instantiated with the compression functions built over ideal block ciphers that are secure against SCA attacks as was done in [36, 17]. If the proposed MAC or hash function mode does not specify any block cipher based compression function then we analyse it using twelve secure compression functions based on block ciphers proposed by Preneel, Govaerts and Vandewalle (PGV) [38]. Such analysis allows the designers to construct hash and DPA resistant MAC modes whose security can be formally reduced to the compression function modes that are real and whose security was formally established [7]. In a related work, HMAC based on the dedicated hash functions SHA-1 and SHA-256 was shown to be vulnerable to the side channel attacks [28,26].

1.1 Our Approach

We analyse several recently proposed provably secure MAC alternatives to the NMAC/HMAC algorithms and NMAC/HMAC settings of some compression function and hash function prototypes proposed in the literature against the DPA attacks. Although the list of MAC algorithms that we have analysed may not be thorough, our analysis can be easily extended to the other similar MAC proposals that we might have missed. The outcome of the DPA attacks on many of these MAC schemes is the recovery of their secret keys. The MAC schemes that are vulnerable to the complete key recovery attacks can be *universally forged*; forgery for any given message. The MACs for which we can partially recover the key or internal state, we can either *existentially forge* the scheme by computing a valid authentication tag for a random message or cannot guarantee its security against forgery attacks. We perform DPA analysis of MACs by dividing them into *Type-1* and *Type-2* categories:

Type-1: Provably secure MAC alternatives to NMAC/HMAC. These MAC schemes, in general, are based on the alternative hash frameworks to the

MD structure. They include BNMAC and its single key variants [43], MAC based on Enveloped Merkle-Damgård (EMD) transform [4], keyed version of Merkle-Damgård with permutation (MDP)-KMDP [18], Multi-Lane NMAC [44] and One-keyed NMAC (O-NMAC) [14]. These schemes do not emphasize using any specific compression function; hence, we analyse their security using twelve secure PGV schemes.

Type-2: NMAC setting of the compression and hash function modes. This category includes the DPA analysis of the NMAC settings of MDC-2 [11,32], MAME [45] and Grindahl [24] compression functions which is also applicable to their HMAC versions. MDC-2 has been chosen for its rigorous analysis and specification in the standards ANSI X9.31 [1] and ISO/IEC 10118-2 [19] and the new proposals Grindahl and MAME are chosen due to their novelty. We assume that the keyed versions of these compression functions define a family of pseudorandom functions and hence their NMAC settings retain the proof of security of NMAC [2]. We also analyse the security of the wide-pipe hash [27] instantiated with twelve PGV schemes in the setting of NMAC.

Our DPA analysis of MACs based on EMD, MDP and wide-pipe hash [27] and the NMAC setting of the hashes in the *Type-2* category meet the criteria of the AHS evaluation process where a MAC version of a secure hash is expected to resist SCA attacks. EMD and KMD modes preserve the pseudorandom oracle and collision resistance properties of the compression functions whereas wide-pipe hash was shown to be secure against the generic attacks of [20,23]. The unkeyed version of O-NMAC was recently shown to be weak against the generic attacks of [23,20] in [15] and collisions can be easily found for the unkeyed version of BNMAC as shown later in the paper.

1.2 Our Results and Their Significance

In Table 1, we outline the results of the DPA attacks on the MAC schemes in the *Type-1* category. While we have analysed MAC functions, where applicable, instantiated with twelve PGV schemes, here we outline our results for the MAC instantiations based on the popular compression functions that include Matyas-Meyer-Oseas (MMO), Miyaguchi-Preneel (MP) and Davies-Meyer (DM). In Table 1, the following notation holds: CK-complete key recovery, PK-partial key recovery, N/A-not applicable, NO-key recovery is not possible, EF-existential forgery, UF-universal forgery, NG-no guarantee on the MAC security. For the sake of comparison, we have also included the results of the DPA analysis of NMAC [36,17] in the last row of Table 1. While MP scheme can be implemented in three different ways for a given block cipher as shown later in the paper, Table 1 outlines the results based on its strongest implementation. The NMAC settings of the compression functions in the *Type-2* category are not vulnerable to the DPA attacks. NMAC based on the wide-pipe hash has similar security as NMAC [36] with respect to the DPA attacks.

Our research indirectly provides some ground work on building DPA resistant hash based MACs using cryptographic primitives such as block ciphers that are often implemented as DPA resistant hardware modules. Our work provides the

Table 1. Our results on the *Type-1* MACs based on three popular PGV schemes

MAC function	Matyas-Meyer-Oseas	Miyaguchi-Preneel	Davies-Meyer
BNMAC [43]	PK(EF)	CK(UF)	CK(UF)
EMD [4]	N/A	N/A	PK(NG)
KMDP [18]	NO	NO	CK(UF)
Multi-lane NMAC [44]	N/A	N/A	PK(NG)
O-NMAC [25]	NO	NO	NO
NMAC [36]	NO	NO	PK(NG)

following significant contributions: Firstly, we analysed several alternatives to NMAC/HMAC that use hash function modes different from that of MD [36,17]. Secondly, we analysed MACs in the MD mode instantiated with the compression functions different from the PGV schemes. Finally, we show the first example of a MAC scheme, in the form of BNMAC, vulnerable to the DPA attacks and can be *existentially forged* when instantiated with any secure compression function.

Our results are the outcome of theoretical DPA attacks on the possible smart card implementations of some MAC proposals. To our knowledge, currently, no smart card implementations of the MAC schemes analysed in this paper are available. So, experimental verification of our analysis has not been done so far. However, since our DPA attacks follow the same attack model used to attack HMAC implementations on an IC chip in [36], it is possible to realise the practicality of our attacks on the MAC implementations on a real smart card system or its emulated system.

1.3 Guide to the Paper

In Section 2, we introduce hash functions and NMAC and HMAC functions. In Section 3, we generalise DPA attacks on MACs. In Sections 4 and 5, we analyse *Type-1* and *Type-2* MAC schemes against the DPA attacks. We conclude the paper in Section 6 with some open questions.

2 Hash Functions

A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ processes an arbitrary length message into a fixed length n-bit hash value. It is a common approach to design H by iterating a compression function $h : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ which processes a fixed length b-bit message and an n-bit input state producing an n-bit output state. The message to be processed using H is always padded using any secure one-to-one padding technique such that $\{0,1\}^* \rightarrow \{0,1\}^{b.t}$ where t is the number of b-bit blocks. The padded message is represented with b-bit message blocks as $m = m[1]\| \dots \|m[t]$. Each block $m[i]$ is processed using h to compute intermediate hash values $H[i] = h_{H[i-1]}(m[i])$ where $i = 1, \dots, t$ and $H[0]$ is the fixed initial value (IV) of H. The final state $H[t] = h_{H[t-1]}(m[t])$ is the hash value of m.

The MD iterative structure [30,12] has been a popular iterated hash function framework used in the design of standard hash functions such as SHA-1 and SHA-2 family [35]. Let e^c be the concatenation of e bit c times where e is either 0 or 1. MD hash functions specify an upper bound of 2^l bits on the length of m and always pad m by appending it with a 1 bit and $0^{b-l-d-1}$ where d is the number of message bits in the incomplete block of m. The last l bits of m are filled in with the binary encoded representation of the length of m in bits (depending on the size of d, an additional block may be used for padding).

Often, compression function modes are constructed using block ciphers. Twelve out of sixty-four PGV compression function modes [38] are provably secure when their underlying block cipher is ideal [7]. This model of PGV uses parameters $p, q, r \in \{H[i-1], m[i], H[i-1] \oplus m[i], 0\}$ and a block cipher G to derive a compression function. See Table 2 for the description of these 12 schemes denoted with h^j where j is from 1 to 12. Table 2 also includes three possible implementations of the compression functions h^3 and h^7. The subscript to G denotes its key input which is either a message block $m[i]$ or a hash state $H[i-1]$ when G is turned into the compression function h using one of the 12 PGV modes. In this paper, we shall often assume that $b = n$ for these twelve PGV schemes.

Table 2. 12 provably secure PGV compression functions

Compression function	Description
h^1	$H[i] = G_{H[i-1]}(m[i]) \oplus m[i]$
h^2	$H[i] = G_{H[i-1]}(m[i] \oplus H[i-1]) \oplus m[i] \oplus H[i-1]$
h^3	$H[i] = G_{H[i-1]}(m[i]) \oplus (m[i] \oplus H[i-1])$
h^4	$H[i] = G_{H[i-1]}(m[i] \oplus H[i-1]) \oplus m[i]$
h^5	$H[i] = G_{m[i]}(H[i-1]) \oplus H[i-1]$
h^6	$H[i] = G_{m[i]}(H[i-1] \oplus m[i]) \oplus (H[i-1] \oplus m[i])$
h^7	$H[i] = G_{m[i]}(H[i-1]) \oplus (m[i] \oplus H[i-1])$
h^8	$H[i] = G_{m[i]}(H[i-1] \oplus m[i]) \oplus H[i-1]$
h^9	$H[i] = G_{H[i-1] \oplus m[i]}(m[i]) \oplus m[i]$
h^{10}	$H[i] = G_{H[i-1] \oplus m[i]}(H[i-1]) \oplus H[i-1]$
h^{11}	$H[i] = G_{H[i-1] \oplus m[i]}(m[i]) \oplus H[i-1]$
h^{12}	$H[i] = G_{H[i-1] \oplus m[i]}(H[i-1]) \oplus m[i]$
$h^{(3,1)}$	$H[i] = (H[i-1] \oplus m[i]) \oplus G_{H[i-1]}(m[i])$
$h^{(7,1)}$	$H[i] = (H[i-1] \oplus m[i]) \oplus G_{m[i]}(H[i-1])$
$h^{(3,2)}$	$H[i] = (G_{H[i-1]}(m[i]) \oplus m[i]) \oplus H[i-1]$
$h^{(7,2)}$	$H[i] = (G_{m[i]}(H[i-1]) \oplus m[i]) \oplus H[i-1]$
$h^{(3,3)}$	$H[i] = (G_{H[i-1]}(m[i]) \oplus H[i-1]) \oplus m[i]$
$h^{(7,3)}$	$H[i] = (G_{m[i]}(H[i-1]) \oplus H[i-1]) \oplus m[i]$

2.1 NMAC and HMAC

Let k_2 and k_1 be any two random and independent secret keys. If H is an MD hash, the NMAC function [3,2] is defined by $\mathrm{NMAC}_{k_1,k_2}(m) = h_{k_1}(H_{k_2}(m))$

where the keys k_2 and k_1 replace the IVs of the inner and outer H where outer H is expected to perform only one iteration. If k is an n-bit random secret key then $\mathrm{HMAC}_k(m) = H_{IV}((k\|0^{b-|k|} \oplus \texttt{const1})\|H_{IV}((k\|0^{b-|k|} \oplus \texttt{const2})\|m)))$. HMAC and NMAC are related by $\mathrm{HMAC}_k(m) = h_{k_1}(H_{k_2}(m))$ where $k_1 = h_{IV}((k\|0^{b-|k|}) \oplus \texttt{const1})$, $k_2 = h_{IV}((k\|0^{b-|k|}) \oplus \texttt{const2})$, `const1` and `const2` are the constants defined in [3] and $\|$ is the concatenation operation.

3 Side Channel Attacks on Hash Based MACs

Here, we generalize the DPA attacks [36, 17] mounted on the NMAC/HMAC algorithms instantiated with the 12 secure PGV compression functions based on the DPA resistant block ciphers.

3.1 Differential Power Analysis (DPA) Attack

The main objective of mounting a DPA attack on a MAC function is to detect target regions in the power consumption of a cryptographic device having a MAC function implementation correlated with particular bits of the secret key.

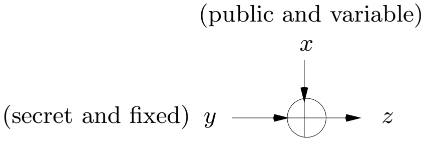


Fig. 1. The DPA attack model

Lemma 1. *A DPA attack is mounted on a MAC function with the target XOR operation $z = x \oplus y$ (See Figure 1) to detect the fixed secret key input y where x is a public variable input.*

The DPA attack on such a MAC function is outlined below:

1. Guess a certain bit b of the secret input y and run the MAC algorithm having the above target XOR operation for N random values of message input x_i where $i = 1 \ldots N$.
2. For each of the N message inputs x_i, a discrete time power signal S_{it} is collected and the corresponding output z_i of the XOR operation is also collected. The index i corresponds to the message input x_i that produced the signal and t corresponds to the time of the sample.
3. Let $x_{i,k}$ be the k^{th} bit of input x_i. Sort the inputs x_i depending on whether the target k^{th} bit of z is 0 or 1. Let $S_b = \{S_{it}|x_{i,k} \oplus b = 0\}$ and $S_{\overline{b}} = \{S_{it}|x_{i,k} \oplus b = 1\}$ where b is the guessed k^{th} bit of y.
4. Compute average power signal for each of the sets S_b and $S_{\overline{b}}$ where $|S_b| + |S_{\overline{b}}| = N$:

$$AP_b[t] = 1/|S_b| \sum_{S_{it} \in S_b} S_{it}$$

$$AP_{\overline{b}}[t] = 1/|S_{\overline{b}}| \sum_{S_{it} \in S_{\overline{b}}} S_{it}$$

5. Following the Hamming weight model of [31], the power consumed by the target XOR operation depends on the Hamming weight of the manipulated data. When there is a large power consumption, that is when $AP_{\overline{b}}[t] \gg AP_b[t]$, the target bit of z is 1 since the other bits behave randomly and the averaging eliminates their effect. This DPA bias signal is used to verify the guess of the secret key bit b of y.
6. By repeating the above steps, the whole secret key y can be recovered. However, it is enough to reclassify the input x and compute average power signal for the new sets using the power signal samples collected in the first instance.

Reverse DPA (RDPA) attack. It is a minor variant of DPA where instead of known input a known output is used.

Lemma 2. *An RDPA attack is mounted on a MAC function with the target XOR operation $z = x \oplus y$ (see Figure 2) to detect the fixed secret key input y where z is a public variable which may not be controlled.*

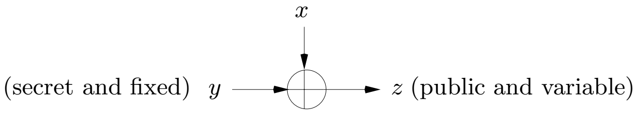


Fig. 2. The RDPA attack model

The RDPA attack on such a MAC function is outlined below:

1. Guess a certain bit of the secret input y and run the MAC algorithm for N random values of input x and collect the discrete time power signals for the XOR operation.
2. Observe the output z for all these N values and sort it out into two groups depending on whether the target bit of input x is 0 or 1.
3. Compute the average power consumption for each group and verify the correctness of the original guess bit of y using the averages.

These results are also applicable when other SCA attacks such as timing or electro magnetic analysis are mounted on the MACs [36].

4 DPA Analysis of *Type-1* Schemes

In this section, we perform DPA analysis of the *Type-1* MAC schemes. Some of these MACs have their own padding rules whereas some others follow the padding functionality defined for the MD hashes.

4.1 BNMAC and Its One-Key Variants

Yasuda [43] proposed BNMAC as an alternative to HMAC to achieve higher performance than HMAC when it is implemented with the slower SHA-2 family. BNMAC uses an alternative hash function framework to the MD called hyper Merkle-Damgård (HMD). Yasuda also proposed two practical variants of BNMAC that use only one secret key.

Hyper Merkle-Damgård. An arbitrary length message m to be processed using a HMD hash function H is split into blocks as $m = m[1] \| m[2] \| \ldots m[2t-1] \| m[2t]$ such that $|m[2i-1]| = n$ and $|m[2i]| = b$ for $i = 1, 2, \ldots, t$. The intermediate hash value $H[i]$ at any iteration i is given by $H[i] = h_{H[i-1] \oplus m[2i-1]}(m[2i])$ where $H[0]$ is the IV of H.

BNMAC and its variants. If k_1 and k_2 are two independent and random secret keys then the BNMAC function is defined by:

$$\mathrm{BNMAC}_{k_1,k_2}(m) = h_{k_1}(H_{k_2}(m) \| 1^{b-n})$$

BNMAC always uses a secure one-to-one padding for m so that the size of m is always a multiple of $b+n$ blocks. The first variant is $\mathrm{BNMAC1}_k(m) = h_{k_1}(H_{k_2}(m) \| 1^{b-n})$ where the two keys k_1 and k_2 are derived from the key k as given by $k_1 = h_k(1^b)$ and $k_2 = h_k(0^b)$. The second variant is $\mathrm{BNMAC2}_k(m) = h_k(H_{h_k(0^b)}(m) \| 1^{b-n})$.

DPA attacks on BNMAC and its variants. By assuming $b = n$, we can instantiate BNMAC with any of the twelve PGV schemes. Then we have $|k_1| = |H_{k_2}(m)|$. In this setting, the first and second set of n input bits to the compression function h correspond to the chaining value and the message block of h. At every iteration i of $\mathrm{BNMAC}_{k_1,k_2}(m)$, the XOR operation $H[i-1] \oplus m[2i-1]$ would become the target on which we mount the DPA attack to recover the previous secret state information $H[i-1]$ where $H[0]$ is the secret key k_2. In fact, this inner key recovery attack on BNMAC is independent of the security of the compression function as the target XOR operation on which we mount the DPA attack is external to the compression function.

Once the key k_2 is recovered, we can mount the DPA attack on the target operation $k_1 \oplus H_{k_2}(m) = \mathrm{BNMAC}_{k_1,k_2}(m)$ based on h^j where $j \in \{2, 4, 6, 8, 9, 10, 11, 12\}$ to recover the secret key k_1. We can mount RDPA attack on this target operation for BNMAC based on h^j where $j \in \{5, (3,1), (3,2), (7,1), (7,2)\}$ to recover the secret key k_1. There is no target XOR operation for BNMAC based on h^j where $j \in \{1, (3,3), (7,3)\}$ on which we could mount the DPA attacks to recover the key k_1. These attacks also apply to BNMAC1 and BNMAC2.

Forging BNMAC. Recovering only k_2, BNMAC based on any secure h can be *existentially forged* with just one oracle query as follows:

1. Query the BNMAC oracle with $m = m[1] \| m[2] \| \ldots m[2t]$ and obtain its tag $\mathrm{BNMAC}_{k_1,k_2}(m)$.

2. Forge BNMAC by computing the tag $\text{BNMAC}_{k_1,k_2}(m^*)$ of the message $m^* = m[1]\|m[2]\|h(m[1] \oplus k_2\|m[2]) \oplus (m[1] \oplus k_2)\|m[2]\| \ldots m[2t]$ such that $\text{BNMAC}_{k_1,k_2}(m^*) = \text{BNMAC}_{k_1,k_2}(m)$.

Note that one can trivially find collisions for the HMD hash. BNMAC invoked with the PGV schemes for which both keys can be recovered can be *universally forged* by computing the tag for any given message.

4.2 Enveloped Merkle-Damgård (EMD) Transform

Bellare and Ristenpart [2] proposed a variant of MD called EMD which works as a MAC when its keyed compression function is a PRF.

EMD construction. An arbitrary length message m to be processed using EMD scheme H is split into b-bit blocks $m[1]\|m[2]\| \ldots m[t-1]$ and incomplete bits are filled in the last block $m[t]$ where $b \geq n + 64$. The last 64 bits of $m[t]$ contain the binary format of $|m|$. At any iteration i of H, the intermediate hash value is given by $H[i] = h_{H[i-1]}(m[i])$ where $1 \leq i \leq t-1$ and $H[0]$ is the IV of H. The hash value is $H[t] = h_{H[0]^*}(H[t-1]\|m[t])$ where $H[0]^*$ is the IV of the final compression function h and $H[0] \neq H[0]^*$. The constraint $b > n$ allows us to use only h^5 as the compression function for the EMD hash.

Keying EMD. The EMD construction keyed through its IVs is defined by $\text{EMD}_{k_1,k_2}(m) = h_{k_1}(H_{k_2}(m[1]\|m[2]\| \ldots m[t-1])\|m[t])$.

Trail secret key-recovery of keyed EMD. We can mount the RDPA attack on the target operation $k_1 \oplus G_{H_{k_2}(m)\|m[t]}(k_1) = \text{EMD}_{k_1,k_2}(m)$ in the outer compression function of EMD MAC based on h^5 to recover the secret key k_1. Note that the control over $b-n-64$-bit input in the block $m[t]$ does not provide us any additional advantage to recover the key k_1. Once we know the key k_1, the current security proof of EMD MAC does not guarantee its security against forgery attacks.

4.3 Merkle-Damgård with Permutation (MDP)

Hirose, Park and Yun [18] proposed a minor variant of the MD called Merkle-Damgård with permutation (MDP). MDP keyed through its chaining value works as a MAC when the underlying keyed compression function is a PRF and secure against a very mild related-key attack.

MDP construction. The MDP construction is obtained by processing the last intermediate hash value of MD using a fixed permutation π. An arbitrary length message m to be processed using MDP is split into b-bit blocks $m[1]\|m[2]\| \ldots \|m[t]$. At any iteration i of the MDP construction, the intermediate hash value is given by $H[i] = h_{H[i-1]}(m[i])$ where $1 \leq i \leq t-1$. The hash value of m is $H[t] = h_{\pi(H[t-1])}(m[t])$. The message m is padded such that the last l bits of $m[t]$ contain $|m|$ in the binary format. We can instantiate MDP using any of the twelve PGV schemes.

Keyed MDP. The MDP scheme keyed through its IV works as a MAC and is called KMDP in [18]. For $1 \leq i \leq t$, the KMDP function is defined by $\mathrm{KMDP}_k(m) = h_{\pi(H[t-1])}(h_k(m[i]))$.

DPA analysis of KMDP. The DPA attack can be mounted on the target XOR operation $H[i-1] \oplus m[i] = H[i]$ of KMDP based on h^j for $j \in \{2, 3(1), 4, 6, 7(1), 8, 9, 10, 11, 12\}$ to recover the secret key k where $H[0] = k$. KMDP based on h^5 is vulnerable to a variant of the RDPA attack as outlined below:

1. Consider a variable 2-block message $m = m[1] \| m[2]$ where the first $b-l-2$ bits of $m[2]$ contain the information followed by the padding bits $1\|0$ and then the binary format of $b+b-l-2$-bit length of the true message in the last l bits of $m[2]$.
2. We repeat the following for N^2 number of random values of m to collect N values of $H[1]$:
   - Choose $m[1]$ and fix it. Mount the RDPA attack on the target operation $\pi(H[1]) \oplus G_{m[2]}(\pi(H[1])) = \mathrm{KMDP}_k(m)$ in the second iteration of KMDP to recover the secret $\pi(H[1])$ by collecting tags $\mathrm{KMDP}_k(m)$ for N values of m by varying the first $b-l-2$ bits of $m[2]$. We then compute $\pi^{-1}(\pi(H[1]))$ to obtain the output $H[1]$ of the first iteration.
3. Finally, we recover the key k by mounting the RDPA attack on the target operation $k \oplus G_{m[1]}(k) = H[1]$ in the first compression function by using N values of $H[1]$ recovered in step 2.

Similarly, RDPA attack can also be mounted on the BNMAC function based on $h^{3(2)}$ and $h^{7(2)}$ to detect the secret key k. In practice, about $N = 100,000 \approx 2^{17}$ samples are required to mount the RDPA attack once on the target operation of a MAC function implemented in an IC chip [36]. Hence, about 2^{35} runs of the compression function of KMDP based on h^j where $j \in \{5, 3(2), 7(2)\}$ implemented on an IC chip are required to recover the secret key.

KMDP based on h^1, $h^{3(3)}$ and $h^{7(3)}$ does not have a target XOR operation on which we could mount the DPA attacks. KMDP based on the PGV compression functions that are vulnerable to the DPA attacks can be *universely* forged for any given message.

4.4 Multilane NMAC

L-lane NMAC and HMAC algorithms. Yasuda [44] proposed a provably secure n-bit L-Lane NMAC ($L \geq 2$), which we call LNMAC, to increase the security level of n-bit NMAC from $2^{n/2}$ to 2^n evaluations against forgery attacks. LNMAC uses L lanes of an MD hash to process an arbitrary length message. Each lane of LNMAC uses an independent random secret key of size n bits as the IV of the hash function in that lane. The proof of security of LNMAC as a PRF and hence as a MAC requires $b \geq 2n$. This condition on b allows us to invoke the LNMAC algorithm with *only* h^5 out of twelve PGV schemes. The 2NMAC function is defined below where $i = 1, 2, \ldots, t$ and τ is the authentication tag:

$$2\text{NMAC}_k(m) = h_{k'}(h_{k'_1}(m[i]) \| h_{k'_2}(m[i]) \| 0^{b-2n}) = \tau$$

Trail secret key-recovery of LNMAC. There is no target XOR operation in 2NMAC based on h^5 on which we can mount the DPA attack to recover the secret keys k'_1 and k'_2. Now let $u = G_{(h_{k'_1}(m[i]) \| h_{k'_2}(m[i]) \| 0^{b-2n})}(k')$. We can mount the RDPA attack on the target operation $k' \oplus u = \tau$ in the last compression function to recover the secret key k'. Similarly, we can recover the key k' for LNMAC as this RDPA attack is independent of the number of lanes. Once we know the trail secret key of LNMAC, its security proof does not guarantee its MAC security.

4.5 O-NMAC

The MAC function O-NMAC was proposed as a one-key variant of NMAC in [14]. O-NMAC computes a linear-XOR checksum using the intermediate hash values of the MD hash function and process it as a final message block. While O-NMAC was analysed informally in [14], a security proof for O-NMAC as a MAC function was provided in [25] under the name Enveloped Checksum Merkle-Damgård (ECM) transform. The O-NMAC function keyed through its IV with a random key k is defined by:

$$\text{O-NMAC}_k(m) = h_{\oplus_{i=1}^{t} h_{H[i-1]}(m[i])}(h_k(m) \| m')$$

where m is split into b-bit blocks $m[1] \| m[2] \| \ldots m[t]$ with the last block $m[t]$ having the binary encoded format of m in its last l bits, $H[0] = k$, $H[i] = h_{H[i-1]}(m[i])$ and m' contains the padding bits including the length encoding of the n-bit value $h_k(m)$ in its last l bits. We assume $|k| = b = n$. Then there is no need to pad $h_k(m)$ with the bits m'. Note that if the message has only one block then a separate block is used to pad it. Hence, at least three iterations of the compression function h are required to compute the authentication tag of an arbitrary length message using O-NMAC.

DPA analysis of O-NMAC. The DPA attack can be mounted on the target XOR operation $H[i-1] \oplus m[i] = H[i]$ of O-NMAC based on h^j for $j \in \{2, 3(1), 4, 6, 7(1), 8, 9, 10, 11, 12\}$ to recover the secret key k where $H[0] = k$. Hence, O-NMAC instantiated with these PGV schemes can be *universally* forged.

When we try to mount the RDPA attack on O-NMAC instantiated with h^5, say using a 2-block message $m = m[1] \| m[2]$, both operands on the left hand side of the expression $H[2] \oplus G_{H[1] \oplus H[2]}(H[2]) = \text{O-NMAC}_k(m)$ are variable. Hence, we cannot mount the RDPA attack. Similar analysis holds for O-NMAC implemented with $h^{3(2)}$ and $h^{7(2)}$. There is no target XOR operation in O-NMAC based on h^j where $j \in \{1, 3(3), 7(3)\}$ on which we could mount the DPA attacks.

5 DPA Analysis of *Type-2* Schemes

In this section, we perform DPA analysis of the NMAC setting of the *Type-2* hash schemes which can also be extended to their HMAC version.

5.1 MDC-2 Hash Function in the NMAC Setting

MDC-2 hash function. MDC-2 [11,32] is a $2n$-bit provably secure hash function based on an n-bit ideal block cipher [39]. MDC-2 is an MD mode of MMO scheme (h^1) in parallel paths. We follow the description of MDC-2 in [39] which is generalised for any ideal block cipher G with the same key and block sizes. If $H[0]$ and $H'[0]$ are two different IVs then the intermediate hash value of MDC-2 at any iteration i is defined by $H[i]\|H'[i] = h^1_{H[i-1]}(m[i])\|h^1_{H'[i-1]}(m[i])$.

MDC-2 hash function in the NMAC setting. Let $k_1 = k_1'\|k_1^*$ and $k_2 = k_2'\|k_2^*$ be any two $2n$-bit keys such that k_1',k_1^*,k_2' and k_2^* are four random and independent keys each of n bits. Then the MDC-2 hash in the NMAC setting is defined by MDC-$2_{k_1,k_2}(m) = h^1_{k_1}(H^1_{k_2}(m))$. At any iteration i, the intermediate hash value of this MAC is given by $H[i]\|H'[i] = h^1_{H[i-1]}(m[i])\|h^1_{H'[i-1]}(m[i])$ where $H[0] = k_1$ and $H'[0] = k_2$.

DPA analysis. There is no target XOR operation in the compression function of MDC-$2_{k_1,k_2}(m)$ on which we could mount the DPA attacks to recover the secret keys.

Remark 1. MDC-4 [8,29] is an extended MDC-2 which uses two sequential executions of MDC-2 to process one message block. The DPA analysis of keyed MDC-2 is also applicable to keyed MDC-4. Similarly, NMAC version of the MD mode of the MAME compression function [45] which uses a novel block cipher algorithm in the MMO mode is also secure against our DPA attacks when its block cipher algorithm is assumed to be ideal and secure against side channel attacks. The MAME compression function was claimed to be transformed to a light weight hash function using any domain extension algorithm and such hash function can withstand side channel attacks when it is used as a key derivation function. We note that not all MAC or key derivative versions of such hash modes may resist DPA attacks even when the block cipher of MAME is ideal. For example, BNMAC based on MAME can be *existentially forged*.

5.2 Grindahl Compression Function in the NMAC Setting

Grindahl compression function design. The Grindahl compression function [24] h processes every block $m[i]$ by concatenating it with the previous state $H[i-1]$ using the permutation G and then truncates the output of G to n-bit state $H[i]$. At any iteration i, its intermediate hash value is defined by $H[i] = h'(G(m[i]\|H[i-1]))$ where h' is the truncation function. When h is iterated in the MD mode, the output of the permutation of the last message block (padded block) is not truncated; instead an output transformation with a pre-defined number of blank rounds is executed followed by the truncation step to output n-bit hash value.

Keying Grindahl compression function. The MD hash iteration of Grindahl compression function can be defined in the NMAC setting using two random and independent secret keys k_2 and k_1. The initial value $H[0]$ of the hash function is replaced with the key k_2 and a key k_1 is used as a key to the outer compression function h. The outer function may require more than one iteration if blank rounds are also defined for the outer function. Its HMAC version can be defined as in HMAC where the two keys k_1 and k_2 are derived from a master key k.

DPA analysis of Keyed Grindahl. There is no target XOR operation in the NMAC setting of Grindahl on which we could mount the DPA attacks when the block algorithm G is ideal. This result complements the claim of [24] on using a side channel resistant AES implementation as the underlying block cipher to protect the keyed implementations of Grindahl members from side channel attacks. Note that the collision attack on Grindahl-256 [37], a specific 256-bit hash function following the design strategy of Grindahl compression function has no influence on our analysis as our analysis assumes an ideal G independent of any specific details.

5.3 Wide-Pipe Hash Construction in the NMAC Setting

Wide-pipe hash. The wide-pipe hash construction [27] uses a large compression function $h : \{0,1\}^{2n} \times \{0,1\}^b \rightarrow \{0,1\}^{2n}$ to process a b-bit message block where $b \geq 2n$ and once the complete message is processed, it uses a function $h' : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ to truncate $2n$-bit output to an n-bit hash value. We assume that the least n significant bits of the output are truncated to produce high order n bits as the hash value.

NMAC with wide-pipe. As noted in [44], a variant of HMAC can be constructed using wide-pipe hash with an n-bit key k. Similarly, we can construct NMAC using wide-pipe hash with two independent n-bit keys k_1 and k_2. Let $k = k_1 \| k_1$ and $k' = k_2 \| k_2$ be two $2n$-bit keys keyed through the IV of the inner/outer wide-pipe hashes of NMAC. We call keyed wide-pipe as WNMAC and define it by $\text{WNMAC}_{k,k'}(m) = h'(h_{k'}(h'(H_k(m))))$.

DPA analysis of WNMAC. For WNMAC based on h^j for $j \in \{2, 3(1), 4, 6, 7(1), 8, 9, 10, 11, 12\}$, the secret key k can be recovered by mounting the DPA attack on the target XOR operation $H[i-1] \oplus m[i] = H[i]$ in the function h^j where $H[0] = k$ as for NMAC. For WNMAC based on h^5, we can mount the RDPA attack on the target XOR operation $G_{h'(H_k(m))}(k') \oplus k' = h_{k'}(h'(H_k(m)))$ to recover the high order n bits of k' which are equal to the tag $\text{WNMAC}_{k,k'}(m)$ and then recover k'. Similarly, we can mount the RDPA attack on WNMAC based on h^j where $j \in \{3(2), 7(2)\}$. There is no target XOR operation in WN-MAC based on h^j where $j \in \{1, 3(3), 7(3)\}$ on which we could mount the DPA attacks.

Remark 2. Protecting a hash based MAC function from the DPA attacks by masking target XOR or addition operations requires developing a whole new hardware module for that MAC function instead of using a widely implemented DPA resistant hardware module of a cryptographic algorithm such as a block cipher. Hence, constructing DPA resistant hash based MACs by using combinations of appropriate key settings and provably secure hash and compression function modes that do not expose any target XOR or addition operations when they are combined with DPA resistant cryptographic hardware modules allows us to reuse these hardware modules.

6 Conclusion

Our research leaves a number of questions open: Among these, the most interesting is, how to design a secure hash mode which can be turned into a DPA resistant provably secure MAC when it is instantiated with any of the secure PGV schemes. The other interesting question is on defining provably secure MAC versions or NMAC settings for some new hash function frameworks such as HAIFA [6] and double-pipe hash [27] invoked with 12 PGV schemes and analyse them against DPA attacks. The final question is how to plug an alternative hash framework to MD into NMAC/HMAC? We believe that our work and future developments in this area of research would provide much needed insights to the designers of hash functions who compete in the AHS process.

Acknowledgments. Many thanks to William Burr and John Kelsey for their encouragement, discussions on the subject and timely responses to our questions regarding SHA-3 evaluation process. We also thank Shoichi Hirose and Krystian Matusiewicz for comments on our paper.

References

1. ANSI. ANSI X9.31:1998: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA). American National Standards Institute (1998)
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.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
4. 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, pp. 299–314. Springer, Heidelberg (2006)

5. Bellovin, S.M., Rescorla, E.K.: Deploying a New Hash Algorithm. In: Proceedings of NDSS. Internet Society (February 2006)
6. Biham, E., Dunkelman, O.: A framework for iterative hash functions - HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007) (Accessed on 5/14/2008), http://eprint.iacr.org/2007/278
7. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
8. Bosselaers, A., Preneel, B.: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040. In: Bosselaers, A., Preneel, B. (eds.) RIPE 1992. LNCS, vol. 1007, pp. 31–67. Springer, Heidelberg (1995)
9. Burr, W.: Personal Communication regarding Frequently Asked Questions on AHS Competition (March 2008)
10. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
11. Coppersmith, D., Pilpel, S., Meyer, C.H., Matyas, S.M., Hyden, M.M., Oseas, J., Brachtl, B., Schilling, M.: Data authentication using modification dectection codes based on a public one way encryption function. U.S. Patent No. 4,908,861, March 13 (1990)
12. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
13. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
14. Gauravaram, P.: Cryptographic Hash Functions: Cryptanalysis, Design and Applications. PhD thesis, Information Security Institute, Queensland University of Technogy (June 2007)
15. Gauravaram, P., Kelsey, J.: Linear-XOR and Additive Checksums Don't Protect Damgård-Merkle Hashes from Generic Attacks. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 36–51. Springer, Heidelberg (2008)
16. Gauravaram, P., McCullagh, A., Dawson, E.: Collision Attacks on MD5 and SHA-1: Is this the "Sword of Damocles" for Electronic Commerce?. In: AusCERT R & D Stream, pp. 1–13 (2006)
17. Gauravaram, P., Okeya, K.: An Update on the Side Channel Cryptanalysis of MACs Based on Cryptographic Hash Functions. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 393–403. Springer, Heidelberg (2007)
18. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
19. ISO/IEC 10118-2. Information Technology - Security Techniques- Hash Functions- Hash functions using an n-bit block cipher. ISO (2000)
20. 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)
21. Kelsey, J.: How Should We Evaluate Hash Submissions?. In: ECRYPT Hash Function Workshop (2007) (Accessed on 02/13/2008), http://csrc.nist.gov/groups/ST/hash/documents/kelsey-ECRYPT2007.pdf

22. Kelsey, J.: How to Choose SHA-3?.In: ECRYPT Hash Function Workshop (2008) (Accessed on 07/26/2008), http://www.lorentzcenter.nl/lc/web/2008/309/presentations/Kelsey.pdf
23. Kelsey, J., Schneier, B.: Second Preimages on n-bit Hash Functions for Much Less than 2n̂ Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
24. Knudsen, L.R., Rechberger, C., Thomsen, S.S.: The Grindahl Hash Functions. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 39–57. Springer, Heidelberg (2007)
25. Lei, D., Chao, L.: Extended Multi-Property-Preserving and ECM-construction. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 361–372. Springer, Heidelberg (2007)
26. Lemke, K., Schramm, K., Paar, C.: DPA on n-bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-Construction. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 205–219. Springer, Heidelberg (2004)
27. Lucks, S.: A Failure-Friendly Design Principle for Hash Functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
28. McEvoy, R.P., Tunstall, M., Murphy, C.C., Marnane, W.P.: Differential power analysis of HMAC based on SHA-2, and countermeasures. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 317–332. Springer, Heidelberg (2008)
29. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography, ch. 9, pp. 321–383. CRC Press, Boca Raton (1997)
30. Merkle, R.: One way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
31. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of power analysis attacks on smartcards. In: Proceedings of the USENIX Workshop on Smartcard Technology, pp. 151–162. USENIX Association (1999)
32. Meyer, C., Schilling, M.: Secure program load with manipulation detection code. In: Proceedings of the 6th Worldwide Congress on Computer and Communications Security and Protection (SECURICOM 1988), Paris, pp. 111–130 (1988)
33. NIST. Federal Information Processing Standard (FIPS PUB 198) The Keyed-Hash Message Authentication Code (HMAC) (March 2002)
34. NIST. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Docket No: 070911510-7512-01 (November 2007)
35. NIST. Federal Information Processing Standard (FIPS PUB 180-3) Secure Hash Standard (2007)
36. Okeya, K.: Side Channel Attacks Against HMACs Based on Block-Cipher Based Hash Functions.. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 432–443. Springer, Heidelberg (2006)
37. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)
38. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
39. Steinberger, J.P.: The collision intractability of MDC-2 in the ideal-cipher model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 34–51. Springer, Heidelberg (2007)

40. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
41. 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)
42. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
43. Yasuda, K.: Boosting Merkle-Damgård Hashing for Message Authentication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 216–231. Springer, Heidelberg (2007)
44. Yasuda, K.: Multilane HMAC - Security beyond the Birthday Limit. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 18–32. Springer, Heidelberg (2007)
45. Yoshida, H., Watanabe, D., Okeya, K., Kitahara, J., Wu, H., Küçük, Ö., Preneel, B.: MAME: A Compression Function with Reduced Hardware Requirements. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 148–165. Springer, Heidelberg (2007)

Key Recovery Attack on Stream Cipher Mir-1 Using a Key-Dependent S-Box

Yukiyasu Tsunoo[1], Teruo Saito[2], Hiroyasu Kubo[2], and Tomoyasu Suzaki[1]

[1] NEC Corporation
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan
{tsunoo@BL,t-suzaki@cb}.jp.nec.com
[2] NEC Software Hokuriku, Ltd.
1, Anyoji, Hakusan, Ishikawa 920-2141, Japan
{t-saito@qh,h-kubo@ps}.jp.nec.com

Abstract. Mir-1 is a stream cipher proposed for Profile 1 at the ECRYPT Stream Cipher Project (eSTREAM). The Mir-1 designer claims a security level of at least 2^{128}, meaning that the secret key cannot be recovered or that the Mir-1 output sequence cannot be distinguished from a truly random number sequence more efficiently than an exhaustive search. At SASC 2006, however, a distinguishing attack on Mir-1 was proposed making use of vulnerabilities in Mir-1 initialization. This paper shows that unknown entries in the key-dependent S-box used by Mir-1 can be classified into partially equivalent pairs by extending the SASC 2006 technique. It also demonstrates an attack that applies that information to recovering the Mir-1 secret key more efficiently than an exhaustive search. To the best of the authors' knowledge, the results described in this paper represent the first successful key recovery attack on Mir-1.

Keywords: eSTREAM, key-dependent S-box, key recovery attack, Mir-1, stream cipher.

1 Introduction

Recently, there have been efforts around the world to standardize encryption. For example, the selection of AES [13] as a standard cipher has been announced in the United States, and the NESSIE project [14] was established in Europe to select standard ciphers. The NESSIE project aimed, in particular, to select secure encryption primitives choosing stream ciphers as one member of that category. However, the stream ciphers offered by the NESSIE project were tackled by numerous cryptanalyses during a 3-year evaluation phase, and ultimately not even a single secure candidate stream cipher remained. In response, the design and analysis of stream ciphers has been receiving increasing attention.

In February 2004, European Network of Excellence for Cryptology (ECRYPT) [5] was established with the objective of encouraging cooperation among European researchers working on information security. The ECRYPT Stream Cipher Project (henceforth "eSTREAM") [4] was established in 2005 by the ECRYPT

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 128–140, 2008.

working group Symmetric Techniques Virtual Lab (STVL), and new stream ciphers were made public. As a result, 34 candidates were submitted to eSTREAM. After more than three years and three phases of evaluation at eSTREAM, 8 stream ciphers have been chosen for the final portfolio.

Mir-1 [10] is a stream cipher proposed at eSTREAM having a 128-bit secret key and a 64-bit initial vector (IV). Although proposed for Profile 1 as a stream cipher for high-speed software implementation, Mir-1 was archived in phase 1. Mir-1 uses a multi-word T-function and a key-dependent S-box function whose security has not been sufficiently studied. Its designer, however, claimed a security level of at least 2^{128}, meaning that the secret key cannot be efficiently recovered or that the Mir-1 output sequence cannot be efficiently distinguished from a truly random number sequence faster than an exhaustive search.

At SASC 2006, however, a distinguishing attack was proposed on Mir-1 [18]. This cryptanalysis method exploits vulnerabilities in T-function characteristics and Mir-1 initialization. The method can distinguish with high probability the output sequence of Mir-1 from a truly random number sequence by choosing only three or four IV pairs. The amount of data needed here is theoretically no more than 2^{10} words. A countermeasure against this technique was also proposed in 2007 [19]. A key recovery attack on Mir-1, however, has not been reported.

In this paper, we show that unknown entries in the key-dependent S-box used by Mir-1 can be classified into partially equivalent pairs by extending the technique of [18]. We also demonstrate an attack for recovering the Mir-1 secret key more efficiently than an exhaustive search based on that information. This attack can recover a secret key with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{110.76}$ table lookups, and a memory complexity of about $2^{32.86}$ bytes. Furthermore, under conditions with no memory limitations, the attack can recover a secret key with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{79.32}$ table lookups, and a memory complexity of about $2^{65.81}$ bytes. We show that Mir-1 incorporating the countermeasure described in [19] is robust to this attack.

To the best of our knowledge, the results described in this paper represent the first successful key recovery attack on Mir-1. We expect the results provided here to be useful in evaluating the security of the many previously proposed block ciphers [2,3,12,16,17] and stream ciphers [1,6,7,9,11,15,20] that use a key-dependent S-box.

Section 2 describes the structure of Mir-1. Section 3 describes a key recovery attack on Mir-1, Section 4 discusses the complexity of this attack and a countermeasure to it, and Section 5 concludes the paper.

2 Description of Mir-1

This section describes the structure of Mir-1, the stream cipher proposed by Maximov at eSTREAM 2005. Ciphertexts are computed by exclusive ORing plaintexts with the keystream generated by the cipher. The keystream generation and initialization of Mir-1 is explained below.

2.1 Notation and Definition

In this paper, bit-wise XOR, AND, and OR are represented by $\oplus$, &, and |, respectively. Addition and multiplication on mod 2^{64} are denoted by $+$ and $\cdot$, respectively. A 64-bit word X rotated to the left by t bits is represented by either $X \lll t$ or $ROL_t(X)$. The byte unit and bit unit of 64-bit word X are set as follows, where $\|$ represents data concatenation.

$$\begin{aligned} X &= X.byte_7 \parallel X.byte_6 \parallel \cdots \parallel X.byte_0 \\ &= X.bit_{63} \parallel X.bit_{62} \parallel \cdots \parallel X.bit_0 \end{aligned}$$

The a^{th} through the b^{th} bits of 64-bit word X are represented by $X[a, b]$. Using the notation described above, we express them as

$$X[a, b] = X.bit_b \parallel X.bit_{b-1} \parallel \cdots \parallel X.bit_a$$

The secret key KEY of Mir-1 is 128-bit long and its initial vector IV is 64-bit long. They are defined as follows.

$$\begin{aligned} KEY &= k_{15} \parallel k_{14} \parallel \cdots \parallel k_0 \\ IV &= IV_7 \parallel IV_6 \parallel \cdots \parallel IV_0 \end{aligned}$$

Here, k_i $(0 \le i \le 15)$ and IV_i $(0 \le i \le 7)$ are one-byte variables.

2.2 Keystream Generation

This section treats Mir-1's keystream generation, which consists of roughly two parts: the loop state update (LS update) and the automata state update (AS update).

The LS update has four registers of 64-bit words $x_i (i = 0, 1, 2, 3)$. Register x_i is updated by a multiword T-function [8]. The LS update is shown in Fig. 1. It guarantees the cycle of maximum length 2^{256}.

The AS update holds two words of 64-bit registers A and B, and it computes A' and B' using the update function shown in Fig. 2. A' and B' correspond to A and B in the next time step. When A' and B' are computed, the register value from the LS update is two 64-bit words obtained by concatenating the upper 32 bits of each of the four registers x_0, x_1, x_2, and x_3. Each 64-bit word is denoted as follows.

$$(x_{i+2}[32, 63] \parallel x_i[32, 63]) \qquad (i = 0, 1)$$

The keystream generation part of Mir-1 performs the LS update and AS update at each clock, and outputs keystream z, that is, the 64-bit B' computed by the AS update.

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 + (s) & + 2 \cdot x_2 \cdot (x_1 \mid C_1) \\ x_1 + (s \,\&\, x_0) & + 2 \cdot x_2 \cdot (x_3 \mid C_3) \\ x_2 + (s \,\&\, x_0 \,\&\, x_1) & + 2 \cdot x_0 \cdot (x_3 \mid C_3) \\ x_3 + (s \,\&\, x_0 \,\&\, x_1 \,\&\, x_2) & + 2 \cdot x_0 \cdot (x_1 \mid C_1) \end{pmatrix}$$

$$\boxed{\begin{array}{c} s = (x_0\&x_1\&x_2\&x_3 + C_0) \oplus x_0\&x_1\&x_2\&x_3 \\ \\ C_0 = \mathrm{0x1248842112488421} \\ C_1 = \mathrm{0x1248124812481248} \\ C_3 = \mathrm{0x4812481248124812} \end{array}}$$

Fig. 1. Loop state update

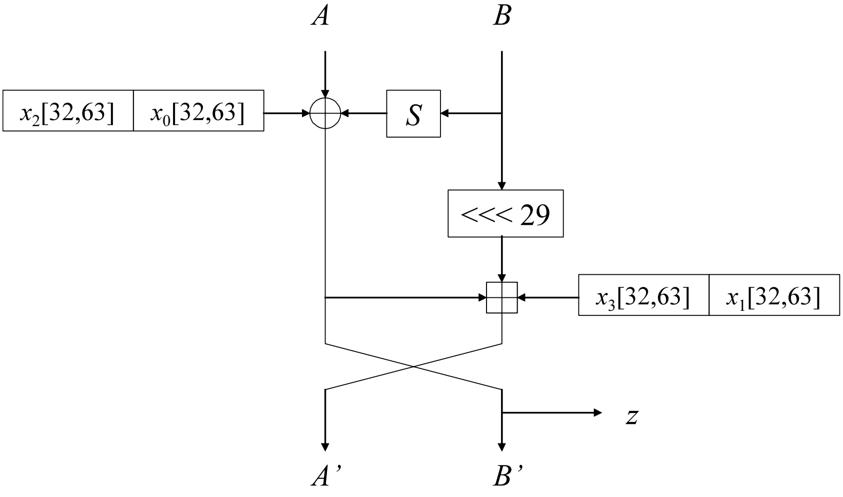


Fig. 2. Automata state update

2.3 Initialization

This section describes Mir-1's initialization part, which also consists of roughly two parts: the key setup and the IV setup.

The key setup initializes register $x_i (i = 0, 1, 2, 3)$ and registers A and B, using a 128-bit secret key. The key setup is shown in Fig. 3.

First, the key setup computes an 8-bit S-box, which varies depending on the secret key value referred to as the secret S-box, using the equation shown below.

```
1. Initialize secret S-box
2. $A = x_1 = (k_7 \| \ldots \| k_0)$
   $B = x_3 = (k_{15} \| \ldots \| k_8)$
   $x_0 = C_0$
   $x_2 = C_1$
3. Repeat 8 times
       Loop State Update
       Automata State Update
```

Fig. 3. Key setup

Here, $SR[\cdot]$ means the S-box of AES. Each entry is computed for $i = 0, \ldots, 255$.

$$S[i] = SR[\cdots SR[SR[i \oplus k_0] \oplus k_1] \oplus \cdots \oplus k_{15}]$$

The IV setup uses a 64-bit initial vector to update register $x_i (i = 0, 1, 2, 3)$ as well as registers A and B. The IV setup is shown in Fig. 4.

3 Key Recovery Attack

This section explains how to apply a key recovery attack against the Mir-1 stream cipher. On applying this attack, it is assumed that the following preconditions are met in an actual situation using stream ciphers.

- The secret key is fixed during the attack.
- Attackers can choose the IV freely.
- Attackers can obtain the keystream generated using the given IV.

3.1 Previous Distinguisher and Extended Distinguisher

This following describes the distinguisher reported in [19] and the distinguisher obtained by extending that distinguisher. To begin with, Theorem 1 holds with respect to the keystream of Mir-1.[1]

Theorem 1. *Given keystreams za and zb generated by IVa ($IV_i = a$, 0x00 $\leq a \leq$ 0xff, $0 \leq i \leq 7$) and IVb ($IV_i = b \neq a$, 0x00 $\leq b \leq$ 0xff, $0 \leq i \leq 7$) that satisfy Eq. (1), Eq. (3) will be satisfied at time t satisfying Eq. (2).*

$$S[a] \equiv S[b] \pmod 2 \tag{1}$$

$$za^{(t)}.byte_0 = zb^{(t)}.byte_0 \tag{2}$$

$$ROL_{29}(za^{(t-1)} \oplus zb^{(t-1)}) \equiv za^{(t+1)} \oplus zb^{(t+1)} \pmod 2 \tag{3}$$

[1] See [19] for proof of Theorem 1.

```
1.  $x_0.byte_4 = x_0.byte_4 \oplus S[IV_0] \oplus S[IV_1] \oplus S[IV_2]$
    $x_1.byte_4 = x_1.byte_4 \oplus S[IV_0] \oplus S[IV_3] \oplus S[IV_4]$
    $x_2.byte_4 = x_2.byte_4 \oplus S[IV_2] \oplus S[IV_5] \oplus S[IV_7]$
    $x_3.byte_4 = x_3.byte_4 \oplus S[IV_3] \oplus S[IV_6] \oplus S[IV_7]$
2.  $x_0.byte_0 = x_0.byte_0 \oplus S[IV_3] \oplus S[IV_5]$
    $x_1.byte_0 = x_1.byte_0 \oplus S[IV_7] \oplus S[IV_6]$
    $x_2.byte_0 = x_2.byte_0 \oplus S[IV_0] \oplus S[IV_1]$
    $x_3.byte_0 = x_3.byte_0 \oplus S[IV_2] \oplus S[IV_4]$
3.  $A.byte_0 = A.byte_0 \oplus S[IV_0] \oplus S[IV_5] \oplus S[IV_6]$
    $A.byte_4 = A.byte_4 \oplus S[IV_1] \oplus S[IV_3] \oplus S[IV_5]$
    $B.byte_0 = B.byte_0 \oplus S[IV_1] \oplus S[IV_4] \oplus S[IV_7]$
    $B.byte_4 = B.byte_4 \oplus S[IV_2] \oplus S[IV_4] \oplus S[IV_6]$
4.  Repeat 2 times
        Loop State Update
        Automata State Update
```

Fig. 4. IV setup

A distinguishing attack using Theorem 1 was proposed in [19]. This attack requires about 2^{10} words at most to distinguish the output sequence of Mir-1 from a truly random number sequence. Theorem 2 below can be easily obtained by extending Theorem 1.

Theorem 2. *Given keystreams za and zb generated by IVa ($IV_i = a$, 0x00 $\leq a \leq$ 0xff, $0 \leq i \leq 7$) and IVb ($IV_i = b \neq a$, 0x00 $\leq b \leq$ 0xff, $0 \leq i \leq 7$) that satisfy Eq. (4), Eq. (6) will be satisfied at time t satisfying Eq. (2) and Eq. (5).*

$$S[a] \equiv S[b] \pmod{2^n, 1 \leq n \leq 7} \tag{4}$$

$$ROL_{29}(za^{(t-1)}) \equiv ROL_{29}(zb^{(t-1)}) \pmod{2^n, 1 \leq n \leq 7} \tag{5}$$

$$za^{(t+1)} \equiv zb^{(t+1)} \pmod{2^n, 1 \leq n \leq 7} \tag{6}$$

Proof. Denoting register x_i updated by IVa and register x_i updated by IVb as xa_i and xb_i, respectively, the condition of Eq. (1) in Theorem 1 can be replaced by Eq. (4) so that register x_i satisfies the following relation.

$$xa_i[0, 31+n] = xb_i[0, 31+n] \quad (0 \leq i \leq 3\ ,\ 1 \leq n \leq 7)$$

This leads to Eq. (7):

$$ROL_{29}(za^{(t-1)} \oplus zb^{(t-1)}) \equiv za^{(t+1)} \oplus zb^{(t+1)} \pmod{2^n, 1 \leq n \leq 7} \tag{7}$$

As a result, substituting Eq. (5) in Eq. (7) gives Eq. (6). □

3.2 Classification of Key-Dependent S-Box

In this section, we describe a method for classifying unknown entries of the key-dependent S-box into partially equivalent pairs using Theorem 2. Specifically, for the case of $n = 7$, the following procedure classifies key-dependent S-box entries into 128 pairs.

1-1. Obtain an N-word keystream for IVa where a is any of 256 values ($IV_i = a$, $0\text{x}00 \leq a \leq 0\text{xff}$, $0 \leq i \leq 7$).[2]

1-2. For keystreams za and zb generated by IVa and IVb ($a < b \leq 0\text{xff}$), check whether Theorem 2 is satisfied for w successive times.[3] If Theorem 2 is satisfied, the IVa and IVb in question are taken to be the 1st pair.

1-3. Repeat step 1-2 for all combinations of (a, b).

The above procedure can classify unknown entries of the key-dependent S-box into 128 pairs whose lower 7 bits are equal. Note, however, that this procedure cannot determine entry values of the key-dependent S-box.

3.3 Key Recovery Method

This section describes a method for recovering a secret key using information on the key-dependent S-box obtained by the method presented in Section 3.2. Specifically, the following procedure - where step 0 signifies a precomputation step - can recover the secret key.

0. For all (x, y) combinations ($0\text{x}00 \leq x < y \leq 0\text{xff}$), store secret-key m-byte $(k_{15-m}, \cdots, k_{15})$ candidates satisfying Eq. (8) in table *tbl*.

$$\begin{aligned} &SR[\cdots SR[x \oplus k_{15-m}] \oplus \cdots \oplus k_{15}] \\ &\quad \equiv SR[\cdots SR[y \oplus k_{15-m}] \oplus \cdots \oplus k_{15}] \pmod{2^7} \end{aligned} \tag{8}$$

Here, the index of *tbl* is a 16-bit value formed by concatenating x and y; it can take on 2^{15} values.

1. Classify key-dependent S-box entries using the method of Section 3.2.
2. Guess l-byte of the secret key $(k_0, \cdots, k_{l-1},\ l = 16 - m)$.
3. Choose the first pair (a, b) obtained in step 1 and calculate (x, y) using the following equations.

$$x = SR[\cdots SR[a \oplus k_0] \oplus \cdots \oplus k_{l-1}]$$
$$y = SR[\cdots SR[b \oplus k_0] \oplus \cdots \oplus k_{l-1}]$$

4. Using the value of (x, y) calculated in step 3 as an index value, obtain a secret-key m-byte candidate $(k_{15-m}, \cdots, k_{15})$ from *tbl*.

[2] Here, N represents the size necessary for classifying key-dependent S-box entries.

[3] Here, w represents the number of times ts needed for classifying key-dependent S-box entries.

5. Choose the 2nd to 128th (a, b) pairs and narrow down the secret-key m-byte $(k_{15-m}, \cdots, k_{15})$ candidates by steps 3 and 4. Once no more candidates remain, guess another l-byte $(k_0, \cdots, k_{l-1})$ of the secret key in step 2 and repeat steps 2 to 5.
6. If a secret-key m-byte $(k_{15-m}, \cdots, k_{15})$ candidate becomes uniquely determined in steps 2 to 5, the 16 bytes that include the l-byte $(k_0, \cdots, k_{l-1})$ guessed in step 2 are taken to be the correct secret key.

Given that the key-dependent S-box can be correctly classified in step 1, the above procedure can recover the secret key more efficiently than an exhaustive search.

4 Discussion

4.1 Complexity of Attack

This section theoretically examines the complexity of each step in the procedure presented in Section 3.3.

First, we investigate the complexity of step 0. The computational complexity T_0 for looking up SR $2m$ times for all combinations of $(x, y, k_{15-m}, \cdots, k_{15})$ is given by

$$\begin{aligned} T_0 &= 2^8 \times 2^7 \times 2^{8m} \times 2m \\ &= m \times 2^{8m+16} \end{aligned}$$

Furthermore, the size of memory M_0 for storing *tbl* is the product of index size, entry size, and number of entries as follows.

$$\begin{aligned} M_0 &= 2^{15} \times m \times 2^{8(m-1)} \\ &= m \times 2^{8m+7} \end{aligned}$$

Next, we examine the complexity of step 1. Since the probability that a keystream that satisfies the conditions of Eqs. (2) and (5) exists is 2^{-15}, keystream generation size is $w \times 2^{15}$ words for one secret-key/IV-pair. Now, to classify the key-dependent S-box into 128 pairs, we consider that the probability of erroneously satisfying Eq. (6) due to variation in the key-dependent S-box should be made sufficiently small, and we therefore set $w = 20$ on the basis of $2^{-128} \gg 2^{-7w}$.[4] Accordingly, the computational complexity T_1 of step 1 is the sum of computational complexity T_1' of step 1-1 and computational complexity T_1'' of steps 1-2 and 1-3 as follows.

$$\begin{aligned} T_1 &= T_1' + T_1'' \\ T_1' &= 2^8 \times 20 \times 2^{15} \\ &= 2^{27.32} \end{aligned}$$

[4] Consider that $2^{-128} \gg \frac{2^{-128}}{1000}$.

$$\begin{aligned}T_1'' &= 20 \times 2^{15} \times (\frac{255}{2} + \frac{253}{2} + \cdots + \frac{1}{2}) \\ &= 2^{18.32} \times \frac{128(1+255)}{2} \\ &= 2^{32.32}\end{aligned}$$

In addition, the amount of required data D_1 and memory M_1 can be given as follows.

$$\begin{aligned}D_1 = M_1 &= 1 \times 2^8 \times 2^{19.32} \times 2^3 \\ &= 2^{30.32}\end{aligned}$$

Now, we examine the complexity of steps 2 to 5. Here, we guess l bytes of the secret key and narrow down the remaining m bytes using *tbl* generated in step 1. If l-byte is correctly guessed, key candidates will be narrowed down to one correct candidate via 128 table lookups. On the other hand, if the l-byte guess is incorrect, key candidates will be narrowed down to one candidate on average on the m^{th} table lookup, and all erroneous key candidates will be rejected on the $(m+1)^{\text{th}}$ table lookup. Computational complexity T_2 for steps 2 to 5 is therefore given as follows.

$$\begin{aligned}T_2 &= 1 \times (2l+1) \times 128 + (2^{8l} - 1) \times (2l+1) \times (m+1) \\ &\approx (2l+1)(m+1) \times 2^{8l}\end{aligned}$$

The complexity of the attack presented in this paper is consequently the sum of the complexities for each of the above steps. Computational complexity T,[5] amount of data D, and memory M are each given as follows:

$$\begin{aligned}T &= T_0 + T_1 + T_2 \\ &= m \times 2^{8m+16} + 2^{27.32} + 2^{32.32} + (2l+1)(m+1) \times 2^{8l} \\ &\approx m \times 2^{8m+16} + (33-2m)(m+1) \times 2^{128-8m} \\ D &= D_1 \\ &= 2^{30.32} \\ M &= M_0 + M_1 \\ &= m \times 2^{8m+7} + 2^{30.32}\end{aligned}$$

Finally, we examine the case for which the attack presented in this paper is most efficient. Assuming no memory limitations, computational complexity T when satisfying

$$m \times 2^{8m+16} \approx (33-2m)(m+1) \times 2^{128-8m}$$

takes on a minimum value for $m = 7$. At this time, computational complexity T is:

$$\begin{aligned}T &\approx 7 \times 2^{72} + 19 \times 8 \times 2^{72} \\ &\approx 2^{79.32}\end{aligned}$$

[5] Although the units of computational complexity differ, the number of table lookups effectively dominates.

In this case, memory M becomes:

$$\begin{aligned} M &= 7 \times 2^{63} + 2^{30.32} \\ &\approx 2^{65.81} \end{aligned}$$

This amount of data cannot, however, be realistically stored. If we therefore assume that the maximum amount of memory that can be realistically provided is 1 terabyte, then, from $M = m \times 2^{8m+7} + 2^{30.32} < 2^{40}$, we get $m = 3$. Computational complexity T in this case turns out to be:

$$\begin{aligned} T &\approx 3 \times 2^{40} + 27 \times 4 \times 2^{104} \\ &\approx 2^{110.76} \end{aligned}$$

Memory M at this time is:

$$\begin{aligned} M &= 3 \times 2^{31} + 2^{30.32} \\ &\approx 2^{32.86} \end{aligned}$$

4.2 Experiment

We performed an experiment to determine whether a secret key could indeed be recovered by the method presented in Section 3.3. The following conditions were established for this experiment.

- Let $(l, m) = (15, 1)$. However, because an exhaustive search across 15 bytes is computationally difficult, the correct secret key was substituted for the 12 bytes $(k_0, \cdots, k_{11})$ and an exhaustive search was performed over the 3 bytes $(k_{12}, \cdots, k_{14})$.[6]
- Letting $w = 20$, a 2^{21}-word keystream was prepared beforehand for one secret-key/IV-pair.
- The probability of success was determined from the results of 100 trials with randomly set secret keys.
- The size of *tbl* generated by the precomputation step was determined.
- The number of *tbl* lookups that dominate computational complexity T was determined.

On performing the experiment based on the above conditions, we successfully classified the key-dependent S-box and derived the secret key for all of the secret keys used in the experiment. The size of *tbl* needed for the precomputation step turned out to be about 2^{15} bytes, which agreed with the theoretical value given in Section 4.1.

On the other hand, the number of *tbl* lookups required for the attack was $2^{28.55}$, which was less than the theoretical value[7] given in Section 4.1. This is because *tbl* generated by the precomputation step is ideally not a table with 1 entry per index value.[8] If we therefore consider the number of *tbl* lookups taking

[6] The 3 bytes used here for performing an exhaustive search can be any 3 bytes from k_0 to k_{14} and not just k_{12} to k_{14}.

[7] $31 \times 2 \times 2^{24} = 2^{29.96}$

[8] Since the SR differential probability is 2^{-6}, the number of entries per index value is divided into 4, 2, and 0.

into account the number of entries per index value in *tbl*, a correctly guessed key would coincide with a correct key obtained on average in 2^{23} attempts in a 24-bit exhaustive search. Here, we have 1 *tbl* lookup (number of entries = 0) at probability 1/2 per index value and 2 *tbl* lookups (number of entries = 2; all candidates are rejected at the 2^{nd} *tbl* lookup) at probability 1/2, so that expectation E of the number of *tbl* lookups needed for the attack would be as follows.

$$E = 2^{23} \times 31 \times \frac{1}{2}(1+2) \approx 2^{28.54}$$

The result obtained by experiment is therefore about the same as the expected value.

4.3 Countermeasure

This section discusses a countermeasure to the attack presented in this paper. The following three structural factors can be given as vulnerabilities in the Mir-1 cipher with respect to this attack.

1. A difference intended for register x_i can be input by choosing the IV.
2. If the difference in the lower n bits of register x_i is 0, it will remain 0 regardless of the number of times LS update is performed.
3. The input value of the key-dependent S-box can be freely chosen by choosing the keystream.

In particular, items 1 and 2 describe vulnerabilities related to IV setup and LS update using the T-function, and item 3 describes a vulnerability related to AS update using the key-dependent S-box. In [19], a distinguishing attack based on vulnerabilities 1 and 2 is proposed along with a countermeasure to that attack. This countermeasure has the following two features.

1. Modifies the method for inserting IV and improves IV setup.
2. Adds bit-permutation processing in which register x_i straddles word boundaries (loop state permutation).

Either of the above enhancements aims to eliminate vulnerabilities in the Mir-1 initialization process. Enhancement 1 aims to prevent a chosen IV attack from being mounted while enhancement 2 aims to spread differential characteristics of the T-function throughout words.

It is shown in [19] that Theorem 1 does not hold as a result of this countermeasure, which means that Theorem 2 likewise does not hold. It can therefore be seen that, in the same way that the distinguishing attack cannot be mounted, neither can classification of the key-dependent S-box be performed. In short, the countermeasure presented in [19] is also robust to the attack presented in this paper.

In addition to the countermeasure proposed in [19], we can consider a means of eliminating the vulnerability in AS update that uses a key-dependent S-box (vulnerability 3 above). Such an enhancement, however, could have a big effect on security with respect to other types of attacks, and for this reason, a careful study must be made. Nevertheless, we have seen that an attack of the kind presented here is capable of recovering the secret key by grouping the entries of the key-dependent S-box in a certain way even though the values of those entries are unknown. Care must therefore be taken when designing a cipher using a key-dependent S-box.

5 Conclusion

In this paper, we showed how unknown entries in the key-dependent S-box used by Mir-1 could be classified into 128 pairs by extending the distinguisher presented in [19] to 7 bits. We described an attack for recovering the Mir-1 secret key more efficiently than an exhaustive search based on that information.

With this method, the secret key can be recovered with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{110.76}$ table lookups, and a memory complexity of about $2^{32.86}$ bytes. And under conditions with no memory limitations, the attack can recover the secret key with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{79.32}$ table lookups, and a memory complexity of about $2^{65.81}$ bytes. We also reported that the countermeasure presented in [19] is effective in resisting this type of attack.

To the best of our knowledge, the results described in this paper represent the first successful key recovery attack on Mir-1. The attack described here can recover the secret key without having to directly determine the values of key-dependent S-box entries. This result shows the possibility of uncovering information on the secret key in ciphers that do not use the key-dependent S-box appropriately. Accordingly, we expect this result to be useful in evaluating the security of block ciphers and stream ciphers that use a key-dependent S-box.

References

1. Anashin, V., Bogdanov, A., Kizhvatov, I., Kumar, S.: ABC: A New Fast Flexible Stream Cipher. eSTREAM submission (2005)
2. Crowley, P.: Mercy: A Fast Large Block Cipher for Disk Sector Encryption. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 49–63. Springer, Heidelberg (2001)
3. Cusick, T.W., Wood, M.C.: The REDOC II Cryptosystem. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 545–563. Springer, Heidelberg (1991)
4. ECRYPT Stream Cipher Project (eSTREAM), http://www.ecrypt.eu.org/stream/
5. European Network of Excellence for Cryptology (ECRYPT), http://www.ecrypt.eu.org/
6. Halevi, S., Coppersmith, D., Jutla, C.S.: Scream: A Software-Efficient Stream Cipher. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 195–209. Springer, Heidelberg (2002)

7. Hawkes, P., Paddon, M., Rose, G.G., de Vries, M.W.: Primitive Specification for SSS. eSTREAM submission (2005)
8. Klimov, A., Shamir, A.: New Cryptographic Primitives Based on Multiword T-functions. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 1–15. Springer, Heidelberg (2004)
9. Li, A.-P.: A New Stream Cipher: Dicing. eSTREAM submission (2005)
10. Maximov, A.: A New Stream Cipher "Mir-1". eSTREAM submission (2005)
11. McGrew, D.A., Fluhrer, S.R.: The Stream Cipher LEVIATHAN. NESSIE submission (2000)
12. Merkle, R.C.: Fast Software Encryption Functions. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 476–501. Springer, Heidelberg (1991)
13. National Institute of Standards and Technology (NIST), Federal Information Processing Standard (FIPS) 197, Advanced Encryption Standard (AES)
14. New European Schemes for Signature, Integrity, and Encryption (NESSIE), https://www.cosic.esat.kuleuven.be/nessie/
15. Rose, G.G., Hawkes, P.: Turing: A Fast Stream Cipher. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 290–306. Springer, Heidelberg (2003)
16. Schneier, B.: Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 191–204. Springer, Heidelberg (1994)
17. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-Bit Block Cipher. NIST AES proposal (1998)
18. Tsunoo, Y., Saito, T., Kubo, H., Shigeri, M.: Cryptanalysis of Mir-1, a T-function Based Stream Cipher. In: Proceedings of SASC 2006, pp. 185–197 (2006), http://www.ecrypt.eu.org/stvl/sasc2006/
19. Tsunoo, Y., Saito, T., Kubo, H., Suzaki, T.: Cryptanalysis of Mir-1: A T-function-Based Stream Cipher. IEEE Transactions on Information Theory 53(11), 4377–4383 (2007)
20. Wu, H.: A New Stream Cipher HC-256. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 226–244. Springer, Heidelberg (2004)

Analysis of Two Attacks on Reduced-Round Versions of the SMS4

Deniz Toz[1] and Orr Dunkelman[2,3]

[1] Middle East Technical University
Institute of Applied Mathematics, Ankara, Turkey
deniz.toz@gmail.com
[2] Katholieke Universiteit Leuven
Department of Electronical Engineering ESAT SDC-COSIC
and
Interdisciplinary Institute for BroadBand Technology (IBBT)
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
orr.dunkelman@esat.kuleuven.be
[3] École Normale Supérieure
Département d'Informatique,
CNRS, INRIA
45 rue d'Ulm, 75230 Paris, France

Abstract. SMS4 is a 128-bit block cipher used in WAPI (the Chinese national standard for wireless networks). Up until recently, the best attacks on SMS4 known, in terms of the number of rounds, were the rectangle attack on 14 rounds and the impossible differential attack on 16 rounds (out of 32 rounds) presented by Lu. While analyzing them, we noticed that these attacks have flaws and that their complexity analysis is inaccurate. In this paper we make a more comprehensive analysis of these attacks and further improve these results.

1 Introduction

SMS4 [1] is a Generalized Feistel Network (GFN) cipher, specified in the Wireless Authentication and Privacy Infrastructure (WAPI), which is mandatory in wireless networks in China. The cipher has block size of 128 bits, and each block is processed in 32 rounds using a secret key of 128 bits long.

The Chinese Standards Association (SAC) submitted WAPI to ISO for recognition as an international standard, at about the same time as the IEEE 802.11i standard. As a result, SMS4 was the subject of an extensive international debate since its introduction. Despite that, up until recently little cryptanalysis of SMS4 was performed. The previously published cryptanalytic results are the differential fault analysis presented in [11], the integral attack on 13 rounds [8], the rectangle attack on 14 rounds and the impossible differential attack on 16 rounds [9], the rectangle attack on 16 rounds and the differential attack on 21 rounds [12], and finally the rectangle attack on 18 rounds, the differential attack and linear attack on 22 rounds [6][1].

[1] We note that the results in [6,12] were found independently of our line of research.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 141–156, 2008.

The cryptanalytic results on SMS4, on which we focus are those of [9]. The proposed rectangle attack on 14 rounds of SMS4 uses $2^{121.82}$ chosen plaintexts and has a claimed time complexity of $2^{116.66}$ 14-round SMS4 computations[2]. The impossible differential attack on 16-round SMS4 from [9] uses 2^{105} chosen plaintexts and its time complexity is conjectured to be 2^{107} 16-round SMS4 computations.

While verifying the results of [9], we found several flaws and possible improvements. In this paper, we show that the actual probability of the 12-round rectangle distinguishers of [9] is $2^{-230.71}$, rather than the claimed probability of $2^{-237.64}$. We also present better 12-round rectangle distinguishers with probability of $2^{-209.78}$. Moreover, we show that the claimed time complexity of the rectangle attack of [9] is flawed due to the deficient process of obtaining candidate quartets, which is not considered in the original time complexity analysis. Therefore, given our improved distinguishers and refined analysis, we present a 14-round rectangle attack that uses $2^{106.89}$ chosen plaintexts pairs and has running time of $2^{107.89}$ encryptions for obtaining the data, $2^{107.89}$ memory accesses to find the pairs, and $2^{87.97}$ encryptions for the analysis. Similarly, we identify several flaws in the impossible differential attack of [9]. We first show that more data is needed than the claimed figures, and then we point out a delicate issue concerning the running time of this attack. We then follow to suggest a corrected attack with data complexity of $2^{117.06}$ chosen plaintexts and time complexity of $2^{117.06}$ encryptions for obtaining the data, $2^{132.06}$ memory accesses for the preliminary elimination, and $2^{95.09}$ encryptions for the analysis.

Independent of our research, SMS4 was analyzed also in [6,12]. A rectangle attack on 16 rounds of SMS4 which requires 2^{124} chosen plaintexts with a time complexity of 2^{116} encryptions, and a differential attack on 21 rounds of SMS4 with data and time complexities of 2^{118} and $2^{112.83}$, respectively are presented in [12]. These results are improved in [6], by using the early abort technique, to a rectangle attack on 18 rounds of SMS4 with a data complexity of 2^{120} and time complexity of $2^{116.83}$, a differential attack on 22 rounds of SMS4 with a data complexity of 2^{118} chosen plaintexts, and a time complexity of $2^{125.71}$ encryptions. Also, a linear attack on 22 rounds of SMS4 which has data complexity of 2^{117} known plaintexts, and time complexity of $2^{109.86}$ encryptions is described in [6].

This paper is organized as follows: In Section 2, we give a brief description of the SMS4 cipher and its properties. In Section 3, we give an overview of the rectangle attack, followed by the previous rectangle attack of [9] on SMS4. Then, we present our observations and improvements for this attack on SMS4. In Section 4, we follow the same outline for the impossible differential attack. Finally, we conclude this paper and summarize our findings in Section 5.

[2] This is the claimed time complexity in [9], but in fact the actual number should be $2^{121.82}$, which is the time required to obtain the ciphertexts to perform attack, according to [9].

2 A Description of SMS4

2.1 Notation

Throughout this paper, we will use the following notation. Each 128-bit *block* is composed of four 32-bit *words* (X_0, X_1, X_2, X_3). Note that the words and blocks are in a "Chinese"-endian order (i.e., the most significant bit is the leftmost bit numbered 0, and the least significant bit is bit 31 for a 32-bit word). Similarly, the most significant byte of a word is the leftmost byte numbered 0, and least significant byte is numbered 3. We denote the bit rotation of the word w by r positions to the left by $w \lll r$; e_j denotes a word whose all positions except the j-th bit are zero and

$$e_{i_1,\ldots,i_j} = e_{i_1} \oplus ... \oplus e_{i_j} \text{ for } 0 \le i_1, \ldots, i_j \le 31$$

2.2 The SMS4 Cipher

SMS4 [1] accepts a 128-bit plaintext $P = (P_0, P_1, P_2, P_3)$ and a 128-bit user key as inputs, and is composed of 32 rounds. In each round, the least significant three bytes of the state are xored with the round key and the result passes the S transformation. The S transformation uses an 8-bit to 8-bit bijective SBox four times in parallel to process each byte, then the concatenated bytes are processed using a linear transformation L. Let $X_i = (X_{i,0}, X_{i,1}, X_{i,2}, X_{i,3})$ and $X_{i+1} = (X_{i+1,0}, X_{i+1,1}, X_{i+1,2}, X_{i+1,3})$ denote the 128-bit input and output to the i-th round, respectively. Then the round function may be formally described by the following equations:

$$\begin{aligned} X_{i+1,0} &= X_{i,1} \\ X_{i+1,1} &= X_{i,2} \\ X_{i+1,2} &= X_{i,3} \\ X_{i+1,3} &= X_{i+1,0} \oplus L(S(X_{i,1} \oplus X_{i,2} \oplus X_{i,3} \oplus RK_i)) \end{aligned}$$

where the S transformation uses the SBox given in [1] and L is the linear transformation:

$$L(x) = x \oplus (x \lll 2) \oplus (x \lll 10) \oplus (x \lll 18) \oplus (x \lll 24) \text{ where } x \in Z_2^{32}$$

The transformation $L \circ S$ is named T in the specification document. RK_i is the 32-bit round sub key for the i-th round, obtained from the key schedule. Decryption is identical to the encryption except for the order of the subkeys, which are used in the reverse order.

Key Schedule: The key schedule is similar to the encryption function. The only difference is that instead of using the linear transformation L, the following linear transformation L' is used:

$$L'(x) = x \oplus (x \lll 13) \oplus (x \lll 23) \text{ where } x \in Z_2^{32}$$

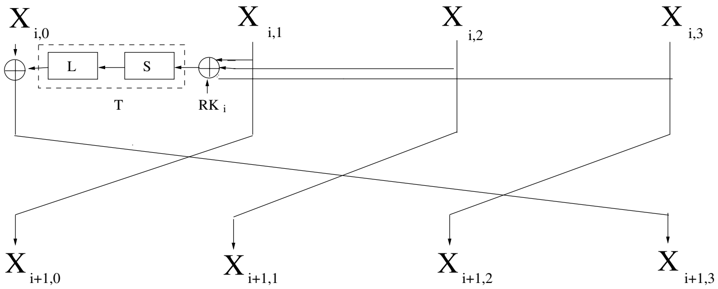


Fig. 1. Round Function

In addition, the user supplied key K is xored with a system parameter, FK. The subkey RK_j of the j-th round is computed as follows:

$$\begin{aligned} FK &= (0xA3B1BAC6, 0x56AA3350, 0x677D9197, 0xB27022DC) \\ k &= (k_0, k_1, k_2, k_3) = K \oplus FK \\ RK_j &= k_{j+4} = k_j \oplus L'(S(k_{j+1} \oplus k_{j+2} \oplus k_{j+3} \oplus CK_j)) \end{aligned}$$

where $CK_j = (ck_{j,0}, ck_{j,1}, ck_{j,2}, ck_{j,3})$ and $ck_{j,k} = 28j + 7k \pmod{256}$.

2.3 Properties and Definitions

Since SMS4 uses a bijective SBox, thus, $S(\Delta x) = 0$ *if and only if* $\Delta x = 0$. The difference distribution table (DDT) of the SBox contains exactly 127 nonzero output differences for a given nonzero input difference. Only one of these values has probability of 2^{-6} while the other 126 remaining nonzero values have probability of 2^{-7}.

The following definitions are used for observing the propagation of any nonzero input difference to the other rounds. In [9], it is not clearly stated to what these sets refer, and the formulas contain typos. Thus, the reader may find the original terminology confusing. Therefore, we rewrite the equations defining these sets, using the same names for the sets (but with a clearer representation).

Given the input difference $(0, e_\Lambda, e_\Lambda, e_\Lambda)$ to the n-th round, where Λ is an arbitrary but nonempty subset of $\{0, 1, \ldots, 31\}$, the set $\theta(e_\Lambda)$ is composed of all the 32-bit differences that an input difference e_Λ to the T function can cause:

$$\theta(e_\Lambda) = \{x | x = L(\Delta d_1), \Pr[S(e_\Lambda) \rightarrow \Delta d_1] > 0 \text{ for } x, e_\Lambda \in Z_2^{32}\}$$

Now, the input difference to the $(n+1)$-th round is $(e_\Lambda, e_\Lambda, e_\Lambda, X)$ where $X \in \theta(e_\Lambda)$. The $\Upsilon(e_\Lambda, X)$ is the set of all 32-bit differences, that an input difference X to the T function may cause after an xor with e_Λ.

$$\Upsilon(e_\Lambda, X) = \{y | y = L(\Delta d_2) \oplus e_\Lambda, \Pr[S(X) \rightarrow \Delta d_2] > 0 \text{ for } X \in \theta(e_\Lambda),\ y, e_\Lambda \in Z_2^{32}\}$$

Similarly, the input difference to the $(n+2)$-th round is of the form $(e_\Lambda, e_\Lambda, X, Y)$ where $X \in \theta(e_\Lambda)$ and $Y \in \Upsilon(e_\Lambda, X)$. The corresponding 32-bit output differences, caused by an input difference $e_\Lambda \oplus X \oplus Y$ to T are denoted by the set

$\Pi(e_\Lambda, X, Y)$.

$$\Pi(e_\Lambda, X, Y) = \{z | z = L(\Delta d_3) \oplus e_\Lambda, \Pr[S(e_\Lambda \oplus X \oplus Y) \rightarrow \Delta d_3] > 0 \text{ for } z, e_\Lambda \in Z_2^{32}\}$$

Finally, the input difference to the $(n+3)$-th round is of the form (e_Λ, X, Y, Z) where $X \in \theta(e_\Lambda)$, $Y \in \Upsilon(e_\Lambda, X)$, and $Z \in \Pi(e_\Lambda, X, Y)$. The set of 32-bit differences after the XOR operation in the $(n+3)$-th round by an input difference $X \oplus Y \oplus Z$ to T is denoted by the set $\Omega(X, Y, Z)$.

$$\Omega(X, Y, Z) = \{w | w = L(\Delta d_4) \oplus e_\Lambda, \Pr[S(X \oplus Y \oplus Z) \rightarrow \Delta d_4] > 0 \text{ for } w \in Z_2^{32}\}$$

3 The Rectangle Attack

The amplified boomerang and rectangle attacks [2] are a chosen plaintext attacks, which evolved from the boomerang attack [10]. The main idea in these attacks is to use two short differential characteristics with high probabilities instead of one long characteristic with a lower probability. The only difference is that the boomerang attack generates a *quartet* at an intermediate value halfway through the cipher, whereas the rectangle attack looks for quartets within a given set of pairs.

For this purpose, the block cipher E is treated as a cascade of two sub-ciphers E_0 and E_1 (i.e, $E = E_1 \circ E_0$). Assume that a differential characteristics $\Delta \rightarrow \Delta^*$ with probability p for E_0, and $\nabla^* \rightarrow \nabla$ with probability q for E_1 are known. The boomerang attack is based on generating right quartets (P_1, P_2, P_3, P_4) which satisfy a set of relations:

1. $P_1 \oplus P_2 = \Delta = P_3 \oplus P_4$.
2. $E_0(P_1) \oplus E_0(P_2) = \Delta^* = E_0(P_3) \oplus E_0(P_4)$.
3. $E_0(P_1) \oplus E_0(P_3) = \nabla^* = E_0(P_2) \oplus E_0(P_4)$.
4. $C_1 \oplus C_3 = \nabla = C_2 \oplus C_4$ where $C_i = E_1(E_0(P_i))$.

A right quartet which satisfies the above equations is formed as follows:

1. Choose a random plaintext P_1 and compute $P_2 = P_1 \oplus \Delta$.
2. Ask for the encryptions of P_1 and P_2 to obtain $C_1 = E(P_1)$ and $C_2 = E(P_2)$.
3. Calculate $C_3 = C_1 \oplus \nabla$ and $C_4 = C_2 \oplus \nabla$.
4. Ask for the decryptions of C_3 and C_4 to obtain $P_3 = D(C_3)$ and $P_4 = D(C_4)$.
5. Check whether $P_3 \oplus P_4 = \Delta$.

The amplified boomerang attack is a chosen plaintext attack in which the same differential conditions have to be satisfied. But instead of generating quartets as given above, a set of plaintext pairs with input difference Δ is generated. Then the aim is to find quartets $((P_1, P_2), (P_3, P4))$ such that $C_1 \oplus C_3 = \nabla = C_2 \oplus C_4$ when $P_1 \oplus P_2 = \Delta = P_3 \oplus P_4$ by using birthday paradox.

By a more careful analysis and a better key recovery algorithm, the amplified boomerang attack was evolved into the rectangle attack. For an optimized method of finding the right rectangle quartet, one may refer to [3].

In [2,10], it is shown that it is possible to use all possible Δ^*'s and ∇^*'s simultaneously. In [2], it is also stated that, if N plaintext pairs with input difference Δ, then the number of expected right quartets is $N^2 2^{-128}\hat{p}^2\hat{q}^2$ for 128-bit block ciphers, where

$$\hat{p} = \sqrt{\sum_{\Delta^*} Pr^2[\Delta \to \Delta^*]} \quad \text{and} \quad \hat{q} = \sqrt{\sum_{\nabla} Pr^2[\nabla^* \to \nabla]}$$

3.1 The Rectangle Attack on 14-Round SMS4 from [9]

The 14-round rectangle attack in [9] uses 12-round rectangle distinguishers with probability[3] $2^{-230.71}$ and requires $2^{121.82}$ chosen plaintexts to attack 14-round SMS4. Let E_0 denote rounds 0 to 7 and let E_1 denote rounds 8 to 11 of SMS4. The differentials used for the 12-round distinguishers of [9] are as follows:

1. **For** E_0: All 8-round differentials of the form $(e_{\psi_1}, e_\psi, e_\psi, e_\psi) \to (e_{\psi_2}, e_{\psi_3}, e_{\psi_4}, e_{\psi_5})$ where only one byte of e_ψ is nonzero and $e_{\psi_1}, e_{\psi_2} \in \theta(e_\psi)$, $e_{\psi_3} \in \Upsilon(e_\psi, e_{\psi_2})$, $e_{\psi_4} \in \Pi(e_\psi, e_{\psi_2}, e_{\psi_3})$, $e_{\psi_5} \in \Omega(e_{\psi_2}, e_{\psi_3}, e_{\psi_4})$, and e_{ψ_1} is fixed.
2. **For** E_1: All 4-round differentials of the form $(e_\Phi, e_\Phi, e_\Phi, 0) \to (e_\Phi, e_\Phi, e_\Phi, e_{\Phi_2})$ where only one byte of e_Φ is nonzero and $e_{\Phi_2} \in \theta(e_\Phi)$.

To calculate the overall probability, the sum of the squares of the probabilities of all used differentials is needed. As there are many 8-round differential characteristics, we list the ones that follow the path in Table 1. In Table 2, we list how many differential characteristics of a given probability follow this path.

Therefore, the lower bound can be calculated as:

$$\begin{aligned}\hat{p}^2 &= (2^{-6})^2 \cdot [(2^{-6})^2 + 126 \cdot (2^{-7})^2] \cdot [(2^{-24})^2 + \tbinom{4}{3} \cdot 126 \cdot (2^{-25})^2 \\ &\quad + \tbinom{4}{2} \cdot 126^2 \cdot (2^{-26})^2 + \tbinom{4}{1} \cdot 126^3 \cdot (2^{-27})^2 + 126^4 \cdot (2^{-28})^2]^3 \\ &= 2^{-102.71}\end{aligned}$$

We note that the second differential is a truncated differential with 127 possible output differences and probability one. Therefore:

$$\hat{q}^2 = 1$$

Thus, the expected number of right rectangle quartets generated by N plaintext pairs is:

$$N^2 \cdot 2^{-128} \cdot \hat{p}^2 \cdot \hat{q}^2 = N^2 \cdot 2^{-230.71}$$

Attack Procedure: The above 12-round distinguishers are used to mount a rectangle attack on 14-round SMS4. Given the 127 input differences[4] $(e_\Phi, e_\Phi,$

[3] In [9] the probability of these 12-round distinguishers is calculated as $2^{-237.64}$ due to a miscalculation of $\hat{q}$.

[4] This is the output difference of the distinguisher, and e_{Φ_2} can take any value (of the 127 possible ones). Thus, the probability of the last step is one. In [9], probability is also calculated for this step, leading to a faulty, lower, probability for the distinguisher.

Table 1. The number of differences and their probabilities for the 8-round characteristic

e_{ψ_1}		e_{ψ_2}		e_{ψ_3}		e_{ψ_4}		e_{ψ_5}	
		(for a fixed e_ψ)		(for a given e_{ψ_2})		(for a given e_{ψ_2}, e_{ψ_3})		(for a given $e_{\psi_2}, e_{\psi_3}, e_{\psi_4}$)	
No	Pr	No	Pr	No	Pr	No	Pr	No	Pr
1	2^{-6}	1	2^{-6}	1	2^{-24}	1	2^{-24}	1	2^{-24}
		126	2^{-7}	$\binom{4}{3} \cdot 126$	2^{-25}	$\binom{4}{3} \cdot 126$	2^{-25}	$\binom{4}{3} \cdot 126$	2^{-25}
				$\binom{4}{2} \cdot 126^2$	2^{-25}	$\binom{4}{2} \cdot 126^2$	2^{-25}	$\binom{4}{2} \cdot 126^2$	2^{-25}
				$\binom{4}{1} \cdot 126^3$	2^{-27}	$\binom{4}{1} \cdot 126^3$	2^{-27}	$\binom{4}{1} \cdot 126^3$	2^{-27}
				126^4	2^{-28}	126^4	2^{-28}	126^4	2^{-28}

Table 2. The number of characteristics and their probabilities for E_0

Probability	2^{-84}	2^{-85}	2^{-86}	2^{-87}	2^{-88}	2^{-89}	2^{-90}
Number	1	$2^{10.678}$	$2^{20.312}$	$2^{29.217}$	$2^{37.514}$	$2^{45.277}$	$2^{52.561}$
Probability	2^{-91}	2^{-92}	2^{-93}	2^{-94}	2^{-95}	2^{-96}	2^{-97}
Number	$2^{59.411}$	$2^{65.872}$	$2^{71.972}$	$2^{77.692}$	$2^{82.920}$	$2^{87.428}$	$2^{90.704}$

$e_\Phi, e_{\Phi_2})$ to round 12, there are 127^5 possible output differences $(e_\Phi, e_\Phi, e_{\Phi_2}, e_{\Phi_3})$ just after round 12, where $e_{\Phi_3} \in \Upsilon(e_\Phi, e_{\Phi_2})$ and 127^9 possible output differences $(e_\Phi, e_{\Phi_2}, e_{\Phi_3}, e_{\Phi_4})$, where $e_{\Phi_4} \in \Pi(e_\Phi, e_{\Phi_2}, e_{\Phi_3})$. For sake of clarity, we define all these output differences by the set Φ:

$$\Phi = \{(e_\Phi, e_{\Phi_2}, e_{\Phi_3}, e_{\Phi_4}) | e_{\Phi_2} \in \Theta(e_\Phi), e_{\Phi_3} \in \Upsilon(e_\Phi, e_{\Phi_2}), e_{\Phi_4} \in \Pi(e_\Phi, e_{\Phi_2}, e_{\Phi_3})\}$$

The proposed attack uses an early abort technique, which allows partially determining whether or not a candidate quartet is a right one by guessing only a small fraction of the subkey, and if not discarding the quartet.

The attack procedure of [9] is as follows:

1. Choose $2^{120.82}$ pairs of plaintexts[5] (P_i, P_i') with input difference $(e_{\psi_1}, e_\psi, e_\psi, e_\psi)$
   (a) Obtain the corresponding ciphertext pairs (C_i, C_i').
   (b) Generate all candidate quartets $((C_{i_1}, C_{i_1}'), (C_{i_2}, C_{i_2}'))$.
   (c) Check whether $C_{i_1} \oplus C_{i_2} \in \Phi$ and $C_{i_1}' \oplus C_{i_2}' \in \Phi$.
2. For each remaining quartet $((C_{i_1}, C_{i_1}'), (C_{i_2}, C_{i_2}'))$:
   (a) For each pair $((C_{i_1}, C_{i_1}')$ and $(C_{i_2}, C_{i_2}'))$ compute the differences in the 4 bytes of their intermediate values just before the L transformation in round 13, and denote them by Δ_{i_1,i_2}^{13} and $\Delta_{i_1,i_2}^{'13}$, respectively. (i.e, compute $\Delta_{i_1,i_2}^{13} = L^{-1}(C_{i_1} \oplus C_{i_2})$ and $\Delta_{i_1,i_2}^{'13} = L^{-1}(C_{i_1}' \oplus C_{i_2}')$.
   (b) For j=0 to 3:

[5] The required number of plaintext pairs is not adapted for the new probability, since it has no effect on our findings.

i. Guess the j-th byte of the subkey RK_{13} and partially decrypt every remaining quartet to obtain the j-th byte of their intermediate values just after the S transformation in round 13. Denote them by $((X_{i_1,j}, X_{i_2,j}), (X'_{i_1,j}, X'_{i_2,j}))$.

ii. Check if $X_{i_1,j} \oplus X_{i_2,j} = \Delta^{13}_{i_1.i_2,j}$ and $X'_{i_1,j} \oplus X'_{i_2,j} = \Delta'^{13}_{i_1.i_2,j}$ and keep only the quartets for which both equalities are satisfied.

3. For each remaining quartet $((T_{i_1}, T'_{i_1}), (T_{i_2}, T'_{i_2}))$ repeat Step 2, for round 12 and RK_{12}.
4. If for a subkey guess (RK_{12}, RK_{13}), there are 6 (or more) remaining quartets try all possible (RK_{10}, RK_{11}) values and perform a trial encryption with one known plaintext/ciphertext pair. If the correct key is not found for all checked (RK_{12}, RK_{13}) values output "failure".

3.2 Improving the 14-Round Attack

By a simple observation, one can conclude that E_0 in the 14-round attack has too many rounds. After round 3, each additional round comes with a cost (in terms of probability) increasing exponentially. Therefore, the attack can be improved by using a shorter characteristic for E_0 with higher probability in exchange for making E_1 longer. We suggest the use of following differential characteristics:

1. **For** E_0: The 6-round differentials $(e_{\psi_1}, e_\psi, e_\psi, e_\psi) \rightarrow (e_\psi, e_\psi, e_{\psi_2}, e_{\psi_3})$ where only one byte of e_ψ is nonzero, $e_{\psi_1}, e_{\psi_2} \in \theta(e_\psi)$, $e_{\psi_3} \in \Upsilon(e_\psi, e_{\psi_2})$, and e_{ψ_1} is fixed.
2. **For** E_1: The 6-round differentials $(e_{\Phi_6}, e_{\Phi_5}, e_\Phi, e_\Phi) \rightarrow (e_\Phi, e_\Phi, e_\Phi, e_{\Phi_2})$ where only one byte of e_Φ is nonzero and $e_{\Phi_5}, e_{\Phi_2} \in \theta(e_\Phi)$, $e_{\Phi_6} \in \Upsilon(e_\Phi, e_{\Phi_5})$.

The details of the rectangle distinguishers for the original attack and the proposed improvement are given in Table 3.

The probability of the new proposed distinguisher can be calculated as follows: As mentioned earlier in Section 3.1, there exists one possible e_{ψ_2} with probability 2^{-6} and 126 possible e_{ψ_2} values with probability 2^{-7} in round 4. And in round 5, for each of the e_{ψ_2} values, we have one possible e_{ψ_3} with probability 2^{-24}, $\binom{4}{1} \times 126$ possible values with probability 2^{-25}, $\binom{4}{2} \times 126^2$ possible values with probability 2^{-26}, $\binom{4}{1} \times 126^3$ possible values with probability 2^{-27} and 126^4 possible values with probability 2^{-28}. Hence for E_0, we have:

$$\begin{aligned}\hat{p}^2 &= (2^{-6})^2 \cdot [(2^{-6})^2 + 126 \cdot (2^{-7})^2] \cdot [(2^{-24})^2 + \tbinom{4}{3} \cdot 126 \cdot (2^{-25})^2 \\ &\quad + \tbinom{4}{2} \cdot 126^2 \cdot (2^{-26})^2 + \tbinom{4}{1} \cdot 126^3 \cdot (2^{-27})^2 + 126^4 \cdot (2^{-28})^2] \\ &= 2^{-46.8881}\end{aligned}$$

Similarly, for E_1, in round 7, we have one possible e_{Φ_5} with probability 2^{-6} and 126 possible e_{Φ_5} with probability 2^{-7}. In round 6, for each of the e_{Φ_5} values, there is one possible e_{Φ_6} with probability 2^{-24}, $\binom{4}{1} \times 126$ possible values with probability 2^{-25}, $\binom{4}{2} \times 126^2$ possible values with probability 2^{-26}, $\binom{4}{1} \times 126^3$

Table 3. The rectangle attack distinguishers

Previous Attack						Improved Attack					
Round	$\Delta X_{i,0}$	$\Delta X_{i,1}$	$\Delta X_{i,2}$	$\Delta X_{i.3}$	Prob	Round	$\Delta X_{i,0}$	$\Delta X_{i,1}$	$\Delta X_{i,2}$	$\Delta X_{i,3}$	Prob
0	e_{ψ_1}	e_ψ	e_ψ	e_ψ	2^{-6}	0	e_{ψ_1}	e_ψ	e_ψ	e_ψ	2^{-6}
1	e_ψ	e_ψ	e_ψ	0	1	1	e_ψ	e_ψ	e_ψ	0	1
2	e_ψ	e_ψ	0	e_ψ	1	2	e_ψ	e_ψ	0	e_ψ	1
3	e_ψ	0	e_ψ	e_ψ	1	3	e_ψ	0	e_ψ	e_ψ	1
4	0	e_ψ	e_ψ	e_ψ	†[a]	4	0	e_ψ	e_ψ	e_ψ	†
5	e_ψ	e_ψ	e_ψ	e_{ψ_2}	†	5	e_ψ	e_ψ	e_ψ	e_{ψ_2}	†
6	e_ψ	e_ψ	e_{ψ_2}	e_{ψ_3}	†	output	e_ψ	e_ψ	e_{ψ_2}	e_{ψ_3}	
7	e_ψ	e_{ψ_2}	e_{ψ_3}	e_{ψ_4}	†	6	e_{Φ_6}	e_{Φ_5}	e_Φ	e_Φ	‡[b]
output	e_{ψ_2}	e_{ψ_3}	e_{ψ_4}	e_{ψ_5}		7	e_{Φ_5}	e_Φ	e_Φ	e_Φ	‡
8	e_Φ	e_Φ	e_Φ	0	1	8	e_Φ	e_Φ	e_Φ	0	1
9	e_Φ	e_Φ	0	e_Φ	1	9	e_Φ	e_Φ	0	e_Φ	1
10	e_Φ	0	e_Φ	e_Φ	1	10	e_Φ	0	e_Φ	e_Φ	1
11	0	e_Φ	e_Φ	e_Φ	1	11	0	e_Φ	e_Φ	e_Φ	1
output	e_Φ	e_Φ	e_Φ	e_{Φ_2}		output	e_Φ	e_Φ	e_Φ	e_{Φ_2}	

[a] The probabilities given with dagger are stated in Table 1.

[b] The probability of e_{Φ_5} is equal to the probability of e_{ψ_2}, since they both belong to the same set. Similarly e_{Φ_6} and e_{ψ_3} have the same probability.

possible values with probability 2^{-27} and 126^4 possible values with probability 2^{-28}. Therefore:

$$\begin{aligned}\hat{q}^2 &= [(2^{-6})^2 + 126\cdot(2^{-7})^2]\cdot[(2^{-24})^2 + \tbinom{4}{3}\cdot 126\cdot(2^{-25})^2 + \tbinom{4}{2}\cdot 126^2\cdot(2^{-26})^2 \\ &+ \tbinom{4}{1}\cdot 126^3\cdot(2^{-27})^2 + 126^4\cdot(2^{-28})^2] \\ &= 2^{-34.8881}\end{aligned}$$

Thus, the expected number of right quartets generated by N plaintext pairs is:

$$N^2\cdot 2^{-128}\cdot\hat{p}^2\cdot\hat{q}^2 = N^2\cdot 2^{-209.78}$$

In order to have sufficient pairs to perform the improved attack $N = 2^{106.89}$ and from this point on we use this figure throughout the analysis.

A flaw in the preliminary elimination: In the original attack of [9], the time complexity is calculated only for candidates of right quartets after the preliminary elimination (the pairs which enter Step 2), and it does not include the time-complexity of the first elimination itself. However, due to the large amount of data, it is impossible to take all the possible pairs and detect candidates for right quartets immediately. We propose the following algorithm for the detection of right quartet candidates:

A more efficient algorithm for the preliminary elimination:

1. Let $C_i = (C_{i,0}, C_{i,1}, C_{i,2}, C_{i,3})$ and $C'_i = (C'_{i,0}, C'_{i,1}, C'_{i,2}, C'_{i,3})$ denote a ciphertext pair.
2. For each ciphertext pair (C_i, C'_i), insert the following entries into a hash table:

   (a) $C_i||C'_i$ to the bin indexed by $C_{i,0}C_{i,1}||C'_{i,0}C'_{i,1}$.
   (b) $C'_i||C_i$ to the bin indexed by $C'_{i,0}C'_{i,1}||C_{i,0}C_{i,1}$.

3. For every $(e_{\Phi_2}, e'_{\Phi_2})$ pair, where $e_{\Phi_2}, e'_{\Phi_2} \in \Theta(e_\Phi)$:

   (a) initialize for each bin a flag to the state of "active".
   (b) For every "active"bin satisfying $C_{i,0}C_{i,1} \leq C'_{i,0}C'_{i,1}$, go to the corresponding bin $C_{i,0}C_{i,1}||C'_{i,0}C'_{i,1} \oplus e_\Phi e_{\Phi_2}||e_\Phi e'_{\Phi_2} = C_{j,0}C_{j,1}||C'_{j,0}C'_{j,1}$.
      i. For all possible combinations of entries $((C_i, C'_i), (C_j, C'_j))$, check whether:
         A. $C_{i,2} \oplus C_{j,2} \in \Upsilon(e_\Phi, e_{\Phi_2})$ and $C'_{i,2} \oplus C'_{j,2} \in \Upsilon(e_\Phi, e'_{\Phi_2})$
         B. $C_{i,3} \oplus C_{j,3} \in \Pi(e_\Phi, e_{\Phi_2}, e_{\Phi_3})$ and $C'_{i,2} \oplus C'_{j,2} \in \Pi(e_\Phi, e'_{\Phi_2}, e'_{\Phi_3})$
         (Once a condition fails, do not check the remaining conditions.)
      ii. Flag the bins $C_{j,0}C_{j,1}||C'_{j,0}C'_{j,1}$ and $C'_{j,0}C'_{j,1}||C_{j,0}C_{j,1}$ as "analyzed".

4. If two pairs satisfying (i)-(ii) are found, keep them as candidates for right quartets, and apply steps 2–4 of the attack in Section 3.1.

If we have N pairs of ciphertexts, the expected number of entries in each of the 2^{128} bins is $2 \cdot N/2^{128} = N \cdot 2^{-127}$ after Step 2. Therefore, we can form $127^2 \cdot (N/2^{127})^2 = N^2/2^{240.02}$ candidate quartets for each pair of bins. Since we are only forming quartets for the bins whose first two words is smaller than its last two words, we analyze 2^{127} pairs of bins. Flagging in Step (b) also prevents analyzing the same quartet twice. So the number of candidate quartets entering Step (i) is $2^{127}/2 \cdot N^2/2^{240.02} = N^2/2^{114.02}$. Now, the probability of passing Step (A) is $(127^4/2^{32})^2 \approx 2^{-8.08}$ and $2^{-8.08} \cdot N^2/2^{114.02} = N^2/2^{122.1}$ quartets remain. The probabilities of Step (A) and (B) are same, thus we have $N^2/2^{130.18}$ candidates for right quartets in Step 3.

The time complexity of the preliminary elimination is as follows: In Step 2, we have $2N$ memory accesses. In step 4, the number of analyzed quartets, which is the number of required memory access is $N^2/2^{114.02}$. Note that there is no need to go over all bins. In total $2N + N^2/2^{114.02}$ memory accesses is required, and thus for $N = 2^{106.89}$, the total running time of the preliminary elimination is expected to be $2^{107.89} + 2^{99.76}$ memory accesses.

For $N = 2^{106.89}$, we have $(2^{106.89})^2/2^{130.18} = 2^{83.6}$ candidates of right quartets. The time complexity of the attack is dominated by the partial decryptions in Step 2(b) for j=0 in [9]. Therefore, the running time of steps 2-4 of the attack is $2^8 \cdot 2^{83.6} \cdot 1/14 = 2^{87.69}$. The total running time is dominated by Step 1, i.e. $2^{107.89}$ memory accesses.

4 Impossible Differential Attack on 16-Round SMS4

4.1 Impossible Differential Attack

Unlike traditional differential cryptanalysis which tracks differences that propagate through the cipher with high probability, impossible differential cryptanalysis exploits differentials with probability zero.

The attack used in [9] is a combination of the general technique called *miss in the middle*, which is used to construct impossible differential, and the *early abort technique* which partially determines whether or not a candidate pair is useful. The main idea is to find two characteristics with probability one, whose conditions cannot be met together [4]. Then, the key can be found by analyzing the rounds surrounding the impossible event, and guessing the subkeys of these rounds. If the impossible event occurs when a candidate key is used, it is obvious that the suggested key is not the right key.

4.2 The Previous Attack on 16-Round SMS4

The attack uses a set of 12-round impossible differentials of the form $(e_\Gamma, e_\Gamma, e_\Gamma, 0) \not\rightarrow (0, e_\Gamma, e_\Gamma, e_\Gamma)$. Two 6-round differentials with probability one are concatenated for the attack. The first differential used in the construction of the impossible differential is $(e_\Gamma, e_\Gamma, e_\Gamma, 0) \rightarrow (e_\Gamma, x_1, y_1, z_1)$ and the second differential is $(z_2, y_2, x_2, e_\Gamma) \rightarrow (0, e_\Gamma, e_\Gamma, e_\Gamma)$, where $x_i \in \Theta(e_\Gamma)$, $y_i \in \Upsilon(e_\Gamma, x_i)$, $z_i \in \Pi(e_\Gamma, x_i, y_i)$ for $i = 1, 2$. These 12-round differentials are used to conduct an impossible differential attack on SMS4 reduced to 16 rounds by adding two additional rounds before and after the differentials.

The attack uses $\Gamma \subseteq \{0, 1, \ldots, 15\}$. Hence, in round 1, for every Γ, there are 127^2 input differences that may lead to e_Γ as the output difference of T, and they can be generated by 127^6 input differences in round 0, which is denoted by the set $\Sigma_1(\Gamma)$ for each Γ. Similarly, there are 127^2 output differences after round 14, that can be generated by e_Γ, and they cause 127^6 possible output differences after round 15 which, is denoted by the set $\Sigma_2(\Gamma)$ for each Γ.[6]

The attack procedure of [9] is as follows:

1. Choose 2^9 structures of 2^{96} plaintexts each where the most significant 2 bytes of the two rightmost words of the plaintexts in each structure is fixed. (Thus, each structure generates $(2^{96})^2/2 = 2^{191}$ plaintext pairs[7] (P_i, P_j) with the desired input difference $(*, *, e_\Gamma, e_\Gamma)$).
   (a) Obtain the corresponding ciphertext pairs of the structures.
   (b) Choose the pairs that satisfy both $P_i \oplus P_j \in \Sigma_1(\Gamma)$ and $C_i \oplus C_j \in \Sigma_2(\Gamma)$ simultaneously for the same Γ, for all possible Γ's.
2. For all the remaining ciphertext pairs (C_i, C_j):

[6] In [9], these sets are denoted by Σ_1 and Σ_2. But actually they are not independent of the choice of Γ. Since the attack procedure runs over all possible such Γ's, it is more clear and more accurate to denote these sets as a function of Γ.

[7] In [9], it was claimed that each structure proposes only 2^{190} plaintext pairs.

Table 4. The two 6-round differentials

Round	$\Delta X_{i.0}$	$\Delta X_{i.1}$	$\Delta X_{i.2}$	$\Delta X_{i.3}$
0	e_Γ	e_Γ	e_Γ	0
1	e_Γ	e_Γ	0	e_Γ
2	e_Γ	0	e_Γ	e_Γ
3	0	e_Γ	e_Γ	e_Γ
4	e_Γ	e_Γ	e_Γ	x_1
5	e_Γ	e_Γ	x_1	y_1
6	e_Γ	x_1	y_1	z_1
6	e_{z_2}	y_2	x_2	e_Γ
7	y_2	x_2	e_Γ	e_Γ
8	x_2	e_Γ	e_Γ	e_Γ
9	e_Γ	e_Γ	e_Γ	0
10	e_Γ	e_Γ	0	e_Γ
11	e_Γ	0	e_Γ	e_Γ
12	0	e_Γ	e_Γ	e_Γ

   (a) Compute the 4-byte difference just before the L transformation in round 15, and denote it by $\Delta^{15}_{i,j}$ (i.e., $\Delta^{15}_{i,j} = L^{-1}(C_{i,3} \oplus C_{j,3})$).
   (b) For l=0 to 3:
      i. Guess the l-th byte of the subkey RK_{15} and partially decrypt (C_i, C_j) to get the l-th byte of the difference just after the S transformation in round 15, denote them by $(T_{i,l}, T_{j,l})$.
      ii. Check if $T_{i,l} \oplus T_{j,l} = \Delta^{15}_{i,j,l}$ and keep the pairs that satisfy the equality.

3. For all the remaining pairs (T_i, T_j):
   (a) Compute the 4-byte difference just before the L transformation in round 14 and denote it by $\Delta^{14}_{i,j}$.
   (b) For l=0 to 1:
      i. Guess the l-th byte of the subkey RK_{14} and partially decrypt (T_i, T_j) to get the l-th byte of their intermediate values just after the S transformation in round 14, denote them by $(Q_{i,l}, Q_{j,l})$.
      ii. Check if $Q_{i,l} \oplus Q_{j,l} = \Delta^{14}_{i,j,l}$ and keep the pairs that satisfy the equality.
4. For all plaintext pairs (P_i, P_j) corresponding to the remaining ciphertexts after Step 3:
   (a) Compute the 4-byte difference just before the L transformation in round 0 and denote by it $\Delta^{0}_{i,j}$.
   (b) For l=0 to 3:
      i. Guess the l-th byte of the subkey RK_0 and partially encrypt (P_i, P_j) to get the l-th byte of their intermediate values just after the S transformation in round 0, denote them by $(R_{i,l}, R_{j,l})$.
      ii. Check if $R_{i,l} \oplus R_{j,l} = \Delta^{0}_{i,j,l}$ and keep the pairs that satisfy the equality.
5. For all the remaining pairs (R_i, R_j):

   (a) Compute the 4-byte difference just before the L transformation in round 1 and denote by $\Delta^1_{i,j}$.
   (b) For l=0 to 1:
      i. Guess the l-th byte of the subkey RK_1 and partially encrypt (R_i, R_j) to get the l-th byte of their intermediate values just after the S transformation in round 1. Denote them by $(S_{i,l}, S_{j,l})$.
      ii. Check if $S_{i,l} \oplus S_{j,l} = \Delta^1_{i,j,l}$. If there exists a qualified pair then discard the guess of 96 subkey bits and try another, otherwise proceed to the next step.
6. Guess the user key from the known subkey values, and perform a trial encryption. If a key is suggested then output it. Otherwise, continue with a new guess of RK_{15} (i.e., go to Step 2).

The claimed time complexity of this attack is of 2^{107} 16-round SMS4 computations in [9] and it requires 2^{105} chosen plaintexts.

4.3 Fixing and Improving the 16-Round Attack

Like in the rectangle attack of [9], in the impossible differential attack of [9], the time complexity analysis is also calculated only for candidates of right pairs after preliminary elimination (the pairs which enter Step 2), and it does not include the time complexity of the first elimination itself. Also, the data complexity suggested in [9] is too low.

Data Complexity Issues: Each structure is composed of 2^{96} plaintexts of the form $(*, *, (a, b_i), (c, d_i))$ where $*$ denotes all possible values and a, c denote the chosen constants of the structure, i.e., each structure suggests $(2^{96})^2/2 = 2^{191}$ pairs. In order to have the desired input difference $(*, *, e_\Gamma, e_\Gamma)$, we must have $b_i \oplus b_j = d_i \oplus d_j = \tilde{e_\Gamma}$ where $\tilde{e_\Gamma}$ is the least significant two bytes of e_Γ for each Γ.

In a structure, for each (b_i, b_j, d_i, d_j), there are $2^{64} \cdot 2^{64} = 2^{128}$ possible pairs of plaintexts. There are $2^{16} \cdot 2^{16}/2 = 2^{31}$ possible (b_i, b_j) pairs, and for each pair there exists 2^{16} possible d_i's. Given (b_i, b_j) and d_i, there exists a unique d_j value satisfying the above condition. Hence, only $2^{128} \cdot 2^{31} \cdot 2^{16} = 2^{175}$ of the 2^{191} pairs satisfy the desired input difference.

$\Sigma_1(\Gamma)$ is composed of $127^6 \simeq 2^{42}$ possible input differences for each Γ. Therefore, the probability of a pair to have $P_1 \oplus P_2 \in \Sigma_1(\Gamma)$ is $2^{42}/2^{64} = 2^{-22}$, and $2^{175} \cdot 2^{-22} = 2^{153}$ pairs pass this step. Note that once the plaintext pair is fixed, Γ is also fixed, so does $\Sigma_1(\Gamma)$ and $\Sigma_2(\Gamma)$. Similar to $\Sigma_1(\Gamma)$, $\Sigma_2(\Gamma)$ is composed of $127^6 \simeq 2^{42}$ possible output differences for each Γ. Therefore, the probability of a pair to have $C_1 \oplus C_2 \in \Sigma_2(\Gamma)$ is $2^{42}/2^{128} = 2^{-84}$ and the number of pairs for a given structure passing the Step 1 of the algorithm is $2^{153} \cdot 2^{-84} = 2^{69}$.

Starting with S such structures, the number of plaintext pairs passing the preliminary elimination is $S \cdot 2^{69}$. The probability that a given subkey is discarded by a given structure is thus, $2^{69} \cdot (2^{-7})^{12} = 2^{-15}$, and that it is not discarded by all S structures is $(1 - 2^{-15})^S$. In order to discard all wrong subkeys, we need to make sure that the probability of a wrong key to remain is about 2^{-96}, i.e., $2^{-96} = 1 - (1 - 2^{-15})^S$. Thus for $S = 2^9$, it is not probable to discard most of the subkey guesses.

The number of required structures can be calculated as follows: There are 2^{96} possible subkeys, and $S \cdot 2^{-15}$ pairs are expected for each subkey. In order to have all wrong subkeys with one pair (i.e., suggested by some pair and thus identified as wrong ones), the probability of a wrong key to have no pairs should be less than 2^{-96}. The probability of having no pairs is $e^{-S \cdot 2^{-15}}$. Solving this, we obtain that $S = 2^{21.06}$ structures are needed for the attack.

Algorithm for the detection of candidate pairs: Denote a plaintext by $P_i = (P_{i,0}, P_{i,1}, (a_i, b_i), (c_i, d_i))$, a ciphertext by $C_i = ((w_i, x_i), (y_i, z_i), C_{i,2}, C_{i,3})$.

1. Insert every plaintext-ciphertext pair (P_i, C_i) of each structure, indexed by the least significant 2 bytes of the rightmost two words of the plaintext and the most significant two words of the corresponding ciphertext (i.e, $b_i||d_i||w_i||x_i||y_i||z_i$) into a hash table.
2. For each $\tilde{e_\varGamma}$:
   (a) For every non-empty bin satisfying $b_i < b_j$:
      i. go to the corresponding bin:
      $b_i||d_i||w_i||x_i||y_i||z_i \oplus \tilde{e_\varGamma}||\tilde{e_\varGamma}||e_\varGamma||e_\varGamma = b_j||d_j||w_j||x_j||y_j||z_j$
      (i.e. $w_i = w_j$ and $y_i = y_j$).
      ii. For all possible combinations of entries, pick the plaintext pairs for which:
         A. $P_{i,1} \oplus P_{j,1} \in \theta(e_\varGamma)$
         B. $C_{i,2} \oplus C_{j,2} \in \theta(e_\varGamma)$
         C. $P_{i,0} \oplus P_{j,0} \in \varUpsilon(e_\varGamma, P_{i,1} \oplus P_{j,1})$
         D. $C_{i,3} \oplus C_{j,3} \in \varUpsilon(e_\varGamma, C_{i,2} \oplus C_{j,2})$
      is satisfied. (If one of them fails, do not check the remaining conditions.)
3. If any pair satisfying (A)-(D) is found, analyze it in Steps 2-6 of the attack.

The time complexity of the preliminary elimination is as follows: In Step 1, we have 2^{96} memory accesses for each structure. There are 2^{96} plaintext-ciphertext pairs in a structure, therefore, the expected number of entries in each of the 2^{96} bins is 1. The resulting number of required memory accesses for Step 2 is $2^{16} \cdot 2^{96}/2 = 2^{111}$ for a given structure. Therefore, the total number of memory accesses of the algorithm is $S \cdot 2^{111} = 2^{132.06}$.

As mentioned earlier in data complexity issues, the number of pairs passing the preliminary elimination is 2^{69} per structure. Therefore, starting with $S = 2^{21.06}$ structures, which results in $2^{117.06}$ chosen plaintexts, the number of plaintext pairs passing the preliminary elimination is $2^{21.06} \cdot 2^{69} = 2^{90.06}$. The time complexity of the partial encryptions/decryptions in Steps 2(b), 3(b), 4(b) and 5(b) of the algorithm is:

$$\sum_{i=1}^{12} \left(2^{91.06} \cdot \frac{1}{127^{i-1}} \cdot 2^8 \right) \cdot \frac{1}{16} = 2^{95.07}$$

However, the time complexity of the attack is dominated by the $2^{117.06}$ partial encryptions required to obtain the ciphertext pairs in Step 1, and by the $2^{132.06}$ memory accesses performed for the preliminary elimination.

5 Summary

In this paper, we reviewed the rectangle attack on 14-rounds and impossible differential attack on 16-rounds SMS4 presented by Lu. We identified some flaws in the attack algorithms and in the time and data complexity analysis of these attacks. We then followed by correcting and improving these attacks.

We first showed that a better 12-round rectangle distinguisher with probability $2^{-209.78}$ can be found, reducing the amount of required chosen plaintexts to perform the attack from $2^{121.82}$ to $2^{107.89}$. Then, we presented a more efficient algorithm to perform the preliminary elimination.

We also identified some flaws in the previous impossible differential attack of [9]. We first showed that more data is needed for the analysis, and we also presented a more efficient algorithm for the preliminary elimination. The results are summarized in Table 5.

Table 5. Comparison of the Results for the Existing Attacks

Attack Type	#of Rounds	Complexity		Source
		Data	Time[a]	
Integral Attack	13	2^{16}	2^{114} Enc	[7]
Rectangle Attack	14	$2^{121.82,b}$	$2^{116.66,b}$ Enc	[9]
Impossible Differential Attack	16	$2^{105,b}$	$2^{107,b}$ Enc	[9]
Rectangle Attack [New]	14	$2^{107.89}$	$2^{107.89}$ MA	Section 3.2
Impossible Differential Attack [New]	16	$2^{117.06}$	$2^{132.06}$ MA	Section 4.3
Rectangle Attack	16	2^{125}	2^{116} Enc	[12]
Boomerang Attack	18	2^{120}	$2^{116.83}$ Enc	[6]
Rectangle Attack	18	2^{124}	$2^{112.83}$ Enc	[6]
Differential Attack	21	2^{118}	$2^{126.6}$ Enc	[12]
Linear Attack	22	2^{117}	$2^{109.86}$ Enc	[6]
Differential Attack	22	2^{118}	$2^{125.71}$ Enc	[6]

Enc - Encryptions, MA - Memory Accesses.

[a] Time complexities are calculated only for the given algorithms, and they do not include the complexity of obtaining the required data for the attack, which may be higher.

[b] As noted in Sections 3.1 and 4.3, these figures are underestimated.

References

1. Beijing Data Security Technology Co. Ltd, *Specification of SMS4* (2006) (in Chinese), http://www.oscca.gov.cn/UpFile/200621016423197990.pdf
2. Biham, E., Dunkelman, O., Keller, N.: The Rectangle Attack Rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
3. Biham, E., Dunkelman, O., Keller, N.: New Results on Boomerang and Rectangle Attacks. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 1–16. Springer, Heidelberg (2002)

4. Biham, E., Birjukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999)
5. Kelsey, J., Kohno, T., Schneier, B.: Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (2001)
6. Kim, T., Kim, J., Hong, S., Sung, J.: Linear and Differential Cryptanalysis of Reduced SMS4 Block Cipher, Cryptology ePrint Archive: Report 2008/281 (2008)
7. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
8. Liu, F., et al.: Analysis of the SMS4 Block Cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007)
9. Lu, J.: Attacking Reduced-Round Versions of the SMS4 Block Cipher in the Chinese WAPI Standart. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 306–318. Springer, Heidelberg (2007)
10. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
11. Zhang, L., Wu, W.: Differential Fault Attack on SMS4. Chinese Journal of Computers 29(9) (2006)
12. Zhang, L., Zhang, W., Wu, W.: Cryptanalysis of Reduced-Round SMS4 Block Cipher. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 216–229. Springer, Heidelberg (2008)

Applying Time-Memory-Data Trade-Off to Meet-in-the-Middle Attack

Jiali Choy, Khoongming Khoo, and Chuan-Wen Loe

DSO National Laboratories
20 Science Park Drive, Singapore 118230
{cjiali,kkhoongm,lchuanwe}@dso.org.sg

Abstract. In this paper, we present several new attacks on multiple encryption block ciphers based on the meet-in-the-middle attack. In the first attack (GDD-MTM), we guess a certain number of secret key bits and apply the meet-in-the-middle attack on multiple ciphertexts. The second attack (TMTO-MTM) is derived from applying the time-memory trade-off attack to the meet-in-the-middle attack on a single ciphertext. We may also use rainbow chains in the table construction to get the Rainbow-MTM attack. The fourth attack (BS-MTM) is defined by combining the time-memory-data trade-off attack proposed by Biryukov and Shamir to the meet-in-the-middle attack on multiple ciphertexts. Lastly, for the final attack (TMD-MTM), we apply the TMTO-Data curve, which demonstrates the general methodology for multiple data trade-offs, to the meet-in-the-middle attack. GDD-MTM requires no pre-processing, but the attack complexity is high while memory requirement is low. In the last four attacks, pre-processing is required but we can achieve lower (faster) online attack complexity at the expense of more memory in comparison with the GDD-MTM attack. To illustrate how the attacks may be used, we applied them in the cryptanalysis of triple DES. In particular, for the BS-MTM attack, we managed to achieve pre-computation and data complexity which are much lower while maintaining almost the same memory and online attack complexity, as compared to a time-memory-data trade-off attack by Biryukov et al. at SAC 2005. In all, our new methodologies offer viable alternatives and provide more flexibility in achieving time-memory-data trade-offs.

Keywords: block cipher, meet-in-the-middle, time-memory-data trade-off, triple DES.

1 Introduction

In [4], Diffie and Hellman described a meet-in-the-middle (MTM) attack that can be applied on a multiple encryption cipher consisting of a concatenation of $L \geq 2$ block ciphers, each with independent keys. Due to this attack, the effective security of triple DES with a 168-bit key is reduced to 112 bits. We give a brief description of the MTM attack in Section 3.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 157–173, 2008.

There were subsequent improvements on the MTM attack after its discovery. One such improvement, described in [6, Section 7.37], is to apply a guess-and-determine technique to the MTM attack. We shall call this attack a guess-and-determine MTM attack (GD-MTM). The GD-MTM attack stipulates that the attacker must initially guess a certain number of key bits of the first block cipher. We propose an extension of this attack, which we call guess-and-determine MTM attack with multiple data (GDD-MTM) to cater for the case when a plaintext is encrypted by the cipher under several keys of which only one needs to be recovered. When applied to triple DES, as compared to the typical MTM attack without pre-computation which needs time complexity of 2^{112} and memory complexity of 2^{56}, the GD-MTM attack achieves lower memory complexity of 2^{48} at the cost of higher time complexity of 2^{120}, while the GDD-MTM attack achieves lower time complexity of 2^{104} at the expense of 2^{14} data and 2^{64} memory requirements. We expound on the GD-MTM and GDD-MTM attacks in Section 4.

In [5], Hellman introduced the time-memory trade-off (TMTO) attack which is a generic cryptanalytic attack that can be applied to any cipher algorithm using one data point. This attack circumvents exhaustive search by pre-computing and pre-storing a large amount of data. During the online phase, the attack uses the data generated by an unknown key to conduct a ciphertext or keystream comparison in order to recover the key quickly. The advantage of this attack is that with pre-computation done offline, the time taken in the online stage is shortened at the expense of more memory required. Later in 2003, Oechslin [8] proposed an improved technique called rainbow chains to derive a single pre-computation table. His method reduces the TMTO attack complexity by a factor of 2.

Then in [3], Biryukov et al. suggested a time-memory-data trade-off attack adapted from [2], which we shall call the BS attack. In the BS attack, they only need to find one key out of a family of ciphertexts which are the encryptions of the same plaintext under different keys. In this way, only a portion of the key space needs to be covered in the pre-computation phase. In addition, they also performed a new unified analysis of Hellman's attack in the presence of multiple data, thereby achieving new time-memory-data trade-offs previously unknown. This paved the way for more flexibility in attack modes depending on case-by-case time-memory-data requirements. The BS attack is a special case of this new framework which we name as the TMD attack.

However, for block ciphers with a large key size such as triple DES, we found that by applying the attacks suggested in [3], the data requirements and pre-computation complexity tend to be high. For example, for the BS attack on triple DES, the data complexity is 2^{42} and the pre-computation complexity is 2^{126}, with 2^{84} complexity for both time and memory. In Sections 5 to 7 of our paper, we address this issue by proposing novel ways of applying the TMTO, Rainbow, BS, and TMD attacks to the MTM attack. We derive four new attacks: the TMTO-MTM, Rainbow-MTM, BS-MTM, and TMD-MTM attacks. While the earlier two sections (on GD-MTM and GDD-MTM attacks) deal with attacks that involve no pre-computation, the last four attacks all require pre-computation.

When applying the TMTO-MTM attack to triple DES using single data, we found that the attack complexity was 2^{80} and the memory complexity was 2^{101}. The pre-processing stage has a complexity of 2^{113}. The Rainbow-MTM attack achieves a slightly better attack complexity of 2^{79} while maintaining the same requirements for memory and data. These can be considered as attacks with attack and memory complexities which are in between the MTM attack without precomputation (2^{112} attack complexity and 2^{56} memory complexity) and the MTM attack with precomputation (2^{56} attack complexity, 2^{113} pre-computation and memory complexities).

The BS-MTM attack is a further generalization of the TMTO-MTM attack in which the goal is now to recover one key out of several keys which are used to encrypt the same plaintext. This attack can be viewed as an improvement of the TMTO-MTM attack as the pre-computation complexity is reduced by a factor equal to the data complexity. Also, if the data, time and memory complexities are denoted by D, M and T respectively, then M^2T is reduced by a factor equal to at least D^2. When applied to triple DES, with pre-computation of 2^{99} and a data complexity of 2^{14}, we can achieve 2^{84} complexity for online attack time and 2^{85} complexity for memory. Thus it has comparable time and memory requirements as Biryukov et al.'s attack [3] but requires less data and pre-computation complexity. This will make the attack more feasible as it is usually very hard to obtain many encryptions (2^{42} compared to 2^{14}) of the same plaintexts under different keys. Moreover, a pre-computation complexity of 2^{99} seems more achievable than a pre-computation complexity of 2^{126}.

More flexibility is achieved by the TMD-MTM attack where we apply the new TMD trade-off in [3] to the MTM attack. For example, if we can afford a larger memory in our previous attack, e.g. 2^{99}, then we can reduce the online attack complexity to 2^{71} while keeping the data complexity at 2^{14} and pre-computation at 2^{99}.

2 Number of Plaintext-Ciphertext Pairs Needed for Verification

Suppose that the B-bit encryption function, E, is the composition of two block ciphers, $E1_{K_1}(\cdot)$ and $E2_{K_2}(\cdot)$, so that E acts on a plaintext P and outputs the ciphertext C given by

$$C = E_{(K_1,K_2)}(P) = E2_{K_2}(E1_{K_1}(P)).$$

Suppose that $K_1 \in GF(2)^{n_1}$, $K_2 \in GF(2)^{n_2}$, and that K_1, K_2 are independent. Then there will be at least $\lfloor \frac{n_1+n_2}{B} \rfloor$ keys mapping a given plaintext P to a certain ciphertext C. Therefore, in order to verify if a possible key (S_1, S_2) is indeed the cipherkey (K_1, K_2), $q = \lceil \frac{n_1+n_2}{B} \rceil$ plaintext-ciphertext pairs

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_q, CT_q)$$

need to be tested to check if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for all i. If so, then $(K_1, K_2) = (S_1, S_2)$.

In the case of multiple data where we only need to find one key out of several keys used to encrypt the same plaintext, the size of the key space is still fixed at $2^{n_1+n_2}$. Therefore, for both cases of single and multiple data, $q = \lceil \frac{n_1+n_2}{B} \rceil$ plaintext-ciphertext pairs are required to verify the correct key.

3 Background: Meet-in-the-Middle Attack (MTM)

The meet-in-the-middle attack is a well-known attack (e.g. see [6, page 235]). We shall give a description of it here. Suppose, as before, that the B-bit encryption function is the composition of two block ciphers so that

$$C = E_{(K_1,K_2)}(P) = E2_{K_2}(E1_{K_1}(P)),$$

where $K_1 \in GF(2)^{n_1}$, $K_2 \in GF(2)^{n_2}$. Assume further, that there are q plaintext-ciphertext pairs available,

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_q, CT_q)$$

where $q = \lceil \frac{n_1+n_2}{B} \rceil$. The meet-in-the-middle (MTM) attack tries to find the key (K_1, K_2) and works according to Algorithm 1 in Appendix A.

Note that if $n_2 < n_1$, we could have stored a table of $C' = E2^{-1}_{K_2}(C)$ in step 1 and then computed $C'' = E1_{K_1}(P)$ to search for collision from the table in step 2. This will take up less memory.

The meet-in-the-middle attack on multiple encryption uses 2^{n_1} memory. The attack complexity is as follows:

1. If $n_1 > B$, then

$$\text{attack complexity} = 2^{n_1} + 2^{n_2} + 2^{n_2+(n_1-B)} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1).$$

The first term corresponds to encryption of PT_1 over possible values of K_1. The middle term corresponds to decryption of CT_1 over possible values of K_2. The magnitude of the last term is due to the fact that for each partially decrypted ciphertext $C'' = E2^{-1}_{K_2}(C_1)$, there will be 2^{n_1-B} keys, S_1, mapping P to C'', so that $2^{n_2+(n_1-B)}$ possible keys will be identified and each has to be verified by $\lceil \frac{n_1+n_2}{B} \rceil - 1$ other plaintext-ciphertext pairs. In this instance, the attack complexity can be subdivided into two cases:
   (a) attack complexity $\approx 2^{n_1+n_2-B}$ if $n_2 > B$; or
   (b) attack complexity $\approx 2^{n_1}$ if $n_2 \leq B$.
2. If $n_1 \leq B$, then

$$\text{attack complexity} = 2^{n_1} + 2^{n_2} + 2^{n_2} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1)$$

since there will be at most one key, S_1, mapping P to each C''. Therefore,
   (a) attack complexity $\approx 2^{n_2}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1}, 2^{n_2})$ if $n_2 \leq B$.

Remark 1. The attack complexity is increased by a factor of at most $\lceil\frac{n_1+n_2}{B}\rceil$ times the approximate complexities we have given in 1(a),(b) and 2(a),(b) above. However, since $\lceil\frac{n_1+n_2}{B}\rceil$ usually lies between 2 and 4, the increase is not significant and hence, we choose not to take it into account in our paper. We will make similar approximations in subsequent derivations.

As can be observed from the discussion above, this attack has complexity much less than the exhaustive search complexity of $2^{n_1+n_2}$ but requires the use of memory.

Example 1. Suppose we apply the MTM attack on triple DES where $E1_{K_1}(\cdot)$ is encryption over the first DES block with a 56-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining two DES blocks with a 112-bit key. This falls into case 2(a) and hence, the attack complexity is 2^{112}. The memory used is 2^{56}. This is less than the exhaustive search complexity of 2^{168}.

Sometimes when we can afford to have high pre-computation complexity and large memory, we can pre-compute step 1 of the MTM attack by fixing a particular plaintext, P, to speed up the actual attack. In this case, $n_1 > n_2$. The attack complexities are similar to cases 1 and 2 described above, except that this time, we exclude the first term, 2^{n_1}, corresponding to the first step in Algorithm 1 since this is done in the pre-computation. In order to ensure that the attack complexity is always given by 2^{n_2}, we shall assume that the key size of K_1 is always bounded by the effective block size by executing the attack on $\lceil\frac{n_1}{B}\rceil$ blocks of B-bit plaintexts, where B is the true block size of the cipher. Then both the memory required and the pre-computation complexity will be increased to $\lceil\frac{n_1}{B}\rceil \cdot 2^{n_1}$ and the effective block size is $B' = B \cdot \lceil\frac{n_1}{B}\rceil$. In this case, only $\lceil\frac{n_1+n_2}{B}\rceil - \lceil\frac{n_1}{B}\rceil$ plaintext-ciphertext pairs are needed for verification of each possible key in step 2 of Algorithm 1. Although we stipulate this requirement only for the MTM attack with pre-computation so far, it shall turn out, as we will see in later sections, that this rule will also apply to all the other attacks in this paper which need pre-computation. Those without pre-computation will not need this requirement. The MTM attack with pre-computation is illustrated in the following example.

Example 2. Suppose we apply the MTM attack on triple DES where $E1_{K_1}(\cdot)$ is encryption over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with a 56-bit key. Since $\lceil\frac{n_1}{B}\rceil = 2$, the effective fixed plaintext is taken to be a concatenation of two 64-bit fixed plaintext blocks. We can pre-compute step 1 of the MTM attack with $2 \cdot 2^{112} = 2^{113}$ complexity and store the result in $2 \cdot 2^{112} = 2^{113}$ memory. In this way, we can recover the key from any ciphertext C encrypted from P with complexity 2^{56} by step 2 of the MTM attack.

Note that in the attack of Example 2, we require the ciphertext to be the encryption of a fixed plaintext. This still works in practice because many plaintext messages contain some fixed header, e.g. MIME header, postscript header, pdf header, version number, and we can take this as PT_1.

To avoid confusion, we shall refer to the attack in Example 1 as a MTM attack without pre-computation and the attack in Example 2 as a MTM attack with pre-computation.

4 Applying Guess-and-Determine Method to Meet-in-the-Middle Attack Using Single and Multiple Data (GD-MTM and GDD-MTM Respectively)

In [6, Section 7.37], there is a suggestion to independently guess s bits of K_1 for some fixed s ($0 \leq s \leq n_1$). In this modification of the MTM attack (without pre-computation), the table for K_1 requires 2^{n_1-s} memory for each set of s bits guessed. We call this version from [6] the GD-MTM attack. As before, suppose that there are q plaintext-ciphertext pairs available,

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_q, CT_q)$$

where $q = \lceil \frac{n_1+n_2}{B} \rceil$. The GD-MTM attack is described in Algorithm 2 in Appendix A.

The memory complexity is 2^{n_1-s} since all computations are done online and the memory is cleared whenever the attacker rebuilds the table for each new guess. The attack complexity is as follows:

1. If $n_1 - s > B$, then

$$\begin{aligned}\text{attack complexity} &= 2^s \left[2^{n_1-s} + 2^{n_2} + 2^{n_2+(n_1-s-B)} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1) \right] \\ &\approx 2^{n_1} + 2^{n_2+s} + 2^{n_1+n_2-B}\end{aligned}$$

   The last term is derived from the fact that for each value of K_1 with s bits fixed and each partially decrypted ciphertext C'', there will be 2^{n_1-s-B} keys, S_1, identified. Again, the attack complexity is given by:
   (a) attack complexity $\approx 2^{n_1+n_2-B}$ if $n_2 > B$; or
   (b) attack complexity $\approx 2^{n_1}$ if $n_2 \leq B$.
2. If $n_1 - s \leq B$, then

$$\begin{aligned}\text{attack complexity} &= 2^s \left[2^{n_1-s} + 2^{n_2} + 2^{n_2} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1) \right] \\ &\approx 2^{n_1} + 2^{n_2+s} + 2^{n_2+s}\end{aligned}$$

   Therefore,
   (a) attack complexity $\approx 2^{n_2+s}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1}, 2^{n_2+s})$ if $n_2 \leq B$.

Comparing the usual MTM attack outlined in Section 3 with the GD-MTM attack, it can be observed that if $n_1 \leq B$ and $n_2 > B$, then the GD-MTM attack is effective for building smaller lookup tables using less memory in exchange for more time needed to repeat the procedure.

We propose a simple extension of GD-MTM to GDD-MTM (guess-and-determine MTM attack with multiple data) for the case where a plaintext is encrypted by the block cipher under several keys and the attacker only needs to recover one key out of several keys. Assume that we have $D = 2^d$ encryptions of P using different keys and we are only required to find one of them, $\left(K_1^{(i)}, K_2^{(i)}\right)$ by attacking:

$$C_0 = E_{(K_1^{(0)},K_2^{(0)})}(P), C_1 = E_{(K_1^{(1)},K_2^{(1)})}(P), \ldots, C_{2^d-1} = E_{(K_1^{(2^d-1)},K_2^{(2^d-1)})}(P)$$

where $E_{(K_1,K_2)}(\cdot) = E1_{K_1}(E2_{K_2}(\cdot))$. Also assume that for each $i = 0, 1, \ldots, 2^{d-1}$, we have q' other plaintext-ciphertext pairs available:

$$(PT_1^{(i)}, CT_1^{(i)}), (PT_2^{(i)}, CT_2^{(i)}), \ldots, (PT_{q'}^{(i)}, CT_{q'}^{(i)}),$$

all encrypted using key $(K_1^{(i)}, K_2^{(i)})$, where $q' = \lceil \frac{n_1+n_2}{B} \rceil - 1$. With multiple data applied to the usual MTM attack without pre-computation described in Algorithm 1, C' only needs to be computed over 2^{n_1-d} values of K_1. However, if we also guess s bits of K_1 at the beginning of the attack, then each table of K_1 will contain just 2^{n_1-d-s} entries. The assumption is that one of the s bits guess should correspond to at least one of the 2^d keys with high probability. The process is outlined in the Algorithm 3 in Appendix A.

In this case, the memory complexity is 2^{n_1-d-s}. The attack complexity is given as follows:

1. If $n_1 - d - s > B$, then

$$\begin{aligned}&\text{attack complexity}\\ &= 2^s \left[2^{n_1-d-s} + 2^d \left(2^{n_2} + 2^{n_2+(n_1-d-s-B)} \cdot (\lceil \tfrac{n_1+n_2}{B} \rceil - 1)\right)\right]\\ &\approx 2^{n_1-d} + 2^{n_2+d+s} + 2^{n_1+n_2-B}\end{aligned}$$

   The two subcases are:
   (a) attack complexity $\approx 2^{n_1+n_2-B}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1-d}, 2^{n_1+n_2-B})$ if $n_2 \leq B$.
2. If $n_1 - d - s \leq B$, then

$$\begin{aligned}&\text{attack complexity}\\ &= 2^s \left[2^{n_1-d-s} + 2^d \left(2^{n_2} + 2^{n_2} \cdot (\lceil \tfrac{n_1+n_2}{B} \rceil - 1)\right)\right]\\ &\approx 2^{n_1-d} + 2^{n_2+d+s} + 2^{n_2+d+s}\end{aligned}$$

   Therefore,
   (a) attack complexity $\approx 2^{n_2+d+s}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1-d}, 2^{n_2+d+s})$ if $n_2 \leq B$.

Example 3. Now let us apply the GD-MTM attack to the 168-bit triple DES encryption of a fixed plaintext P where $E1_{K_1}(\cdot)$ is encryption of a fixed plaintext P over the first DES block with a 56-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining two DES blocks with a 112-bit key. Assume that our system has a memory limitation of 2^{48}, that is, $s = 8$. Then the time complexity is $2^{112+8} = 2^{120}$.

Example 4. In this example, we shall apply the GDD-MTM attack to 168-bit triple DES encryption of a fixed plaintext P to recover one key out of a key-pool of 2^{14} keys, that is, $d = 14$. We take $E1_{K_1}(\cdot)$ to be encryption of a fixed plaintext P over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ to be encryption over the remaining DES block with a 56-bit key. Suppose we can afford 2^{64} memory so that s is fixed at 34. Then the time complexity is given by $\max(2^{112-14}, 2^{56+14+34}) = 2^{104}$.

Comparing Examples 3 and 4, we see that with multiple data, the attacker can achieve lower time complexity at the expense of more memory requirement.

5 Applying Time-Memory Trade-Off to Meet-in-the-Middle Attack (TMTO-MTM and Rainbow-MTM)

In this section, we apply the time-memory trade-off (TMTO) attack of [5] to the MTM attack with pre-computation and we call it the TMTO-MTM attack. Basically, we apply the TMTO attack to the pre-computation step 1 of the MTM attack. Again we have the same restriction as Example 2, i.e. we require the ciphertext to be the encryption of a fixed plaintext P. Furthermore, for this attack and subsequent ones (Rainbow-MTM, BS-MTM, and TMD-MTM), all of which require pre-computation, we shall also assume that the key size of K_1 is always bounded by the effective block size, B'. This is done by attacking $\lceil \frac{n_1}{B} \rceil$ blocks of B-bit plaintexts, where B is the effective block size of the cipher. Then $B' = B \cdot \lceil \frac{n_1}{B} \rceil$ and both the memory and pre-computation complexities are increased by a factor of $\lceil \frac{n_1}{B} \rceil$. Let E' be defined by

$$E'_{(K_1,K_2)}(P_1 \parallel \ldots \parallel P_{\lceil \frac{n_1}{B} \rceil}) = E2_{K_2}(E1_{K_1}(P_1)) \parallel \ldots \parallel E2_{K_2}(E1_{K_1}(P_{\lceil \frac{n_1}{B} \rceil})).$$

$E1'$ and $E2'$ are also defined in a similar way.

For the TMTO-MTM attack, assume that we have $C = E'_{(K_1,K_2)}(P)$ (where P is a concatenation of $\lceil \frac{n_1}{B} \rceil$ fixed plaintext blocks). Also assume that we have q'' other plaintext-ciphertext pairs available:

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_{q''}, CT_{q''}),$$

all encrypted with E using key (K_1, K_2), where $q'' = \lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil$. The algorithm is shown in Algorithm 4 in Appendix A.

As we have noted, the pre-processing complexity in step 1 is $2^{n_1} \cdot \lceil \frac{n_1}{B} \rceil = mt^2 \cdot \lceil \frac{n_1}{B} \rceil$. Assuming we use a memory of $2^{mem} \cdot \lceil \frac{n_1}{B} \rceil = mt \cdot \lceil \frac{n_1}{B} \rceil$ for our attack, the attack complexity in step 2 is

$$\begin{aligned}
&t^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (mt^2/mt)^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (2^{n_1}/2^{mem})^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= 2^{2(n_1-mem)+n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&\approx 2^{2(n_1-mem)+n_2} + 2^{n_2}.
\end{aligned}$$

Example 5. Suppose as in Example 2, we apply the TMTO-MTM attack on triple DES where $E1_{K_1}(\cdot)$ is encryption over the first two DES blocks with 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with 56-bit keys. The effective fixed plaintext is taken to be a concatenation of two 64-bit fixed plaintext blocks. Suppose we can only afford 2^{101} instead of 2^{113} memory. Then the pre-processing complexity of TMTO-MTM is 2^{113} but the attack complexity is now $2^{2(112-100)+56} = 2^{80}$ instead of 2^{56}.

Thus the TMTO-MTM attack is a trade-off of less memory at the expense of more attack complexity for the pre-computed MTM attack.

In [8], Oechslin published a new way of forming the pre-computation table using rainbow chains. In this method, only one table is required with mt starting points and chains of length t, where $mt^2 =$ *size of key space*. Each chain uses t reduction functions, f_i, starting with reduction function f_1 and ending with reduction function f_t. If two chains collide, they merge only if the collision appears at the same position in both chains. The probability of success of the rainbow attack was found to be approximately equal to the success probability of t classical tables of size $m \times t$. This method reduces the number of table look-ups by a factor of t compared to the Hellman's original TMTO method. Rainbow chains eliminate the occurrence of loops and any merges amongst chains are detectable. Furthermore, they reduce the total complexity of the attack by a factor of 2.

We may apply the rainbow attack to the precomputation step of the MTM attack in a way analogous to how the TMTO-MTM attack was constructed. We call this the Rainbow-MTM attack. More specifically, the process is given in Algorithm 5.

So for this attack, as compared to the TMTO-MTM attack, the time complexity is reduced slightly to $2^{2(n_1-mem)+n_2-1} \cdot \lceil \frac{n_1}{B} \rceil + 2^{n_2}$ while the memory and pre-computation requirements remain the same.

Remark 2. In [1, Section 6.1], there was a criticism levelled against rainbow chains. According to them, in the Hellman scheme, it is possible to store just half the number of bits of the start and end points compared to the Rainbow scheme, reducing time complexity by a factor of 4 and therefore offsetting the claimed improvement of the Rainbow scheme which only reduces the time complexity by a smaller factor of 2. However, we believe that rainbow chains still offer other important advantages as given in [8, Section 3].

Remark 3. In [7], the authors suggested another method of dealing with the case when the key size is greater than the block size of the cipher. For their attack, the pre-processing stage computes 2^{n_1-B} tables, where each row of a particular table starts with a B-bit string which is then concatenated with an (n_1-B)-bit fixed string to encrypt the zero vector. The result is then concatenated with the fixed string again and the same procedure is repeated to obtain a chain of values. Note, however, that they assume the more stringent requirement that $B < n_1 \leq 2B$.

6 Applying Biryukov-Shamir Time-Memory-Data Trade-Off to Meet-in-the-Middle Attack (BS-MTM)

Let us consider a scenario where a plaintext is encrypted by a block cipher under several keys. One such example was suggested by Biryukov et al. in [3] where they attacked the Unix password encryption scheme. Because they only need to recover one key out of several keys, they can apply the time-memory-data (TMD) trade-off attack of [2]. To distinguish between this attack and a more general attack that we will discuss in the next section, we will refer to this attack as the BS attack. In the BS attack, the complexity of pre-processing can be reduced from N to N/D where N is the key space and D is the size of the keypool from which one needs to be found.

In this section, we shall try to recover the key from ciphertexts encrypted from a fixed plaintext P by applying the BS attack to the MTM attack with pre-computation. Our new attack is called the BS-MTM attack. Assume, as in Section 4 that we have $D = 2^d$ encryptions of P (where P is a concatenation of $\lceil \frac{n_1}{B} \rceil$ fixed plaintext blocks) using different keys and we only need to find one of them, i.e. find one of $(K_1^{(i)}, K_2^{(i)})$ by attacking:

$$C_0 = E'_{(K_1^{(0)},K_2^{(0)})}(P), C_1 = E'_{(K_1^{(1)},K_2^{(1)})}(P), \ldots, C_{2^d-1} = E'_{(K_1^{(2^d-1)},K_2^{(2^d-1)})}(P),$$

where $E'_{(K_1,K_2)}(\cdot) = E1'_{K_1}(E2'_{K_2}(\cdot))$. This means that the effective block size to be $B' = B \cdot \lceil \frac{n_1}{B} \rceil \geq n_1$. Also assume that for each $i = 0, 1, \ldots, 2^{d-1}$, we have q'' other plaintext-ciphertext pairs available:

$$(PT_1^{(i)}, CT_1^{(i)}), (PT_2^{(i)}, CT_2^{(i)}), \ldots, (PT_{q''}^{(i)}, CT_{q''}^{(i)}),$$

all encrypted with E using key $(K_1^{(i)}, K_2^{(i)})$, where $q'' = \lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil$. Algorithm 6 in Appendix A illustrates the attack.

As we have noted, the pre-processing complexity in step 1 is $2^{n_1-d} \cdot \lceil \frac{n_1}{B} \rceil$. Assuming we use a memory of $2^{mem} \cdot \lceil \frac{n_1}{B} \rceil = mt/2^d \cdot \lceil \frac{n_1}{B} \rceil$ for our attack, the attack complexity in step 2 is

$$\begin{aligned}
&2^d \cdot t^2/2^d \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^d \cdot 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (mt^2/mt)^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (2^{n_1}/(2^{mem} \times 2^d))^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= 2^{2(n_1-d-mem)+n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&\approx 2^{2(n_1-d-mem)+n_2} + 2^{n_2+d}.
\end{aligned}$$

Example 6. Suppose we want to attack the 168-bit triple DES encryption of a fixed plaintext P and we only need to recover one key out of a key pool of 2^{14} keys. We apply the BS-MTM attack where $E1_{K_1}(\cdot)$ is encryption of a fixed plaintext P over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with a 56-bit key. If we use 2^{85} memory, then the pre-computation complexity is $2^{112-14} \cdot 2 = 2^{99}$ and the attack complexity is $2^{2(112-14-84)+56} = 2^{84}$. In comparison, according to [3, Table 2], a direct application of the time-memory-data trade-off attack on triple DES where we recover one key out of a pool of 2^{42} keys requires: 2^{84} memory, 2^{126} pre-computation and 2^{84} attack complexity. Therefore, our BS-MTM attack achieves lower data and pre-processing complexity than the BS attack.

7 Applying TMTO - Data Curve to Meet-in-the-Middle Attack (TMD-MTM)

In [3], the authors also presented a unifying framework for the analysis of multiple data trade-offs. The BS attack adapted from [2] is considered as a special case of this more general framework. Furthermore, they identified a new class of single table multiple data trade-offs which cannot be obtained from the BS attack. In this section, we shall apply their more general TMD attack to the MTM attack with pre-computation and we call our new attack the TMD-MTM attack. The mode of this attack follows a similar procedure to the BS-MTM attack as outlined in Algorithm 6. The main difference lies in the use of the birthday bound as will be highlighted later. Suppose, as before, that $B' = B \cdot \lceil \frac{n_1}{B} \rceil \geq n_1$. The attack is given by Algorithm 7 in Appendix A.

It is now easy to see that the BS-MTM attack is a special case of the TMD-MTM attack with $z = 1$, i.e. $mt^2 = N$. In Example 6, we used the parameters $(w, x, y, z) = (\frac{1}{8}, 1, \frac{3}{4}, 1)$.

Example 7. Now let us apply the TMD-MTM attack to the 168-bit triple DES encryption of a fixed plaintext P to recover one key out of a keypool of 2^{14} keys. In this case $w = \frac{1}{8}$. As before, $E1_{K_1}(\cdot)$ is encryption of a fixed plaintext P over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with a 56-bit key. In order to achieve minimum attack complexity $T' = 2^{56} \cdot N^{x-y} + 2^{70}$, we take x to be minimum (i.e. $x = 1$) and y to be maximum (i.e. $y = 1 - w = \frac{7}{8}$). Then $z = 3 - (2w + x + y) = \frac{7}{8}$. This gives $M' = 2^{99}$, $PC' = 2^{99}$ and $T' = 2^{71}$ with parameters $(w, x, y, z) = (\frac{1}{8}, 1, \frac{7}{8}, \frac{7}{8})$. Since $r = x = 1$, only 1 table is needed. Comparing with the BS-MTM attack in Example 6, this TMD-MTM attack has lower attack complexity at the expense of more memory required.

8 Conclusion

In this paper, we presented one new no pre-computation attack using the guess-and-determine technique for multiple data — the GDD-MTM attack — improvising the previously known GD-MTM attack for single data. We have also proposed four new attacks involving pre-computation — the TMTO-MTM, Rainbow-MTM, BS-MTM, and TMD-MTM attacks — by applying the TMTO, Rainbow, BS, and TMD attacks respectively to the MTM attack. Figure 1 in Appendix B gives a comparison of the time-memory-data trade-offs of the attacks on triple DES.

Remark 4. For all the attacks with pre-computation in Figure 1 except the BS attack, the attacks are done on two blocks of fixed plaintext.

As can be observed, the attacks without pre-computation generally have higher attack complexities but require less memory. The attacks with pre-computation can afford much lower online attack complexities but require larger memory and pre-computations. Another difference is that the no pre-computation attacks are known plaintext attacks while the pre-computation attacks are chosen plaintext attacks. In both types of attacks, the presence of multiple data can help reduce the time/memory/pre-computation complexities.

Our proposed attacks provide viable methods to achieve new time-memory-data trade-offs apart from previously known attacks. In particular, our new attacks involving pre-computation are desirable as they can achieve lower data and pre-computation complexity than the attacks suggested by Biryukov [3].

References

1. Barkan, E., Biham, E., Shamir, A.: Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 1–21. Springer, Heidelberg (2006)
2. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Trade-offs for Stream Ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000)
3. Biryukov, A., Mukkhopadhyay, S., Sarkar, P.: Improved Time-Memory Trade-Off with Multiple Data. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 110–127. Springer, Heidelberg (2006)
4. Diffie, W., Hellman, M.: Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10(6), 74–84 (1977)
5. Hellman, M.: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans. on Information Theory 26, 401–406 (1980)
6. Menezes, A., van Oorshot, P.C., Vanstone, S.: Handbook of Applied Cryptography, ch. 7. CRC Press, Boca Raton (1996)
7. Mihaljevic, M., Fossorier, M., Imai, H.: Security Evaluation of Certain Broadcast Encryption Schemes Employing a Generalized Time-Memory-Data Trade-off. IEEE Communication Letters 11, 988–990 (2007)
8. Oechslin, P.: Making a Faster Cryptanalytic Time-Memory Trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003)

A Algorithms

Algorithm 1: MTM Attack

1. *Compute* $C' = E1_{K_1}(PT_1)$ *over all possible values of* K_1*. Store the pair* (C', K_1) *and sort according to* C'*. This step has complexity* 2^{n_1} *and needs* 2^{n_1} *memory.*
2. *Compute* $C'' = E2_{K_2}^{-1}(CT_1)$ *over all possible values of* K_2*. For all* K_2*, look for a match for* C'' *from the pair* (C'', K_2) *in the stored table. Once any possible key* (S_1, S_2) *has been identified, test if* $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ *for*

all $i = 2, \ldots, q$. Discard (S_1, S_2) if it does not satisfy this equation for any i. After this step, all the wrong keys will be filtered out, leaving only the correct key with overwhelming probability.

Algorithm 2: GD-MTM Attack

1. *Fix s bits of K_1 at a particular value.*
2. *Compute $C' = E1_{K_1}(PT_1)$ over all values of K_1 with the s bits fixed as stated in step 1. The table for K_1 will only contain 2^{n_1-s} entries. This step has 2^{n_1-s} time complexity and requires 2^{n_1-s} memory.*
3. *Compute $C'' = E2^{-1}_{K_2}(CT_1)$ over all possible values of K_2. For all K_2, look for a match for C'' from the pair (C'', K_2) in the table. Once any possible key (S_1, S_2) has been identified, test if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for all $i = 2, \ldots, q$. Discard (S_1, S_2) if it does not satisfy this equation for any i.*
4. *Repeat steps 2 and 3 by varying the same s bits over all possible values.*

Algorithm 3: GDD-MTM Attack

1. *Fix s bits of K_1 at a particular value.*
2. *Compute $C' = E1_{K_1}(P)$ over 2^{n_1-d-s} values of K_1 with the s bits fixed as stated in step 1. The table for K_1 will only contain 2^{n_1-d-s} entries. This step has time complexity 2^{n_1-d-s} and requires 2^{n_1-d-s} memory.*
3. *Compute $C''_0 = E2^{-1}_{K_2}(C_0)$ over all possible values of K_2. For all K_2, look for a match for C''_0 from the pair (C''_0, K_2) in the table. Once any possible key (S_1, S_2) has been identified, test if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for $\lceil \frac{n_1+n_2}{B} \rceil - 1$ other plaintext-ciphertext pairs (PT_i, CT_i). Discard (S_1, S_2) if it does not satisfy this equation for any i.*
4. *If the correct key $(K_1^{(0)}, K_2^{(0)})$ is not found, then repeat steps 3 and 4 for $j = 1, \ldots, 2^d - 1$ consecutively until one correct key is found. Note that the set of $\lceil \frac{n_1+n_2}{B} \rceil - 1$ plaintext-ciphertext pairs used for verification need not be uniform across all the different C_j.*
5. *If no correct key is found by step 4, vary the same s bits of K_1 over the rest of the 2^s possibilities and repeat steps 1 to 4 until a correct key is found.*

Algorithm 4: TMTO-MTM Attack

1. *Pre-processing:*
   (a) *Choose two positive integers m, t such that $2^{n_1} = mt^2$. Fix a plaintext P (where P is a concatenation of $\lceil \frac{n_1}{B} \rceil$ plaintext blocks) and let $C = E2'_{K2}(E1'_{K1}(P))$. Define a one-way function $g(K) = E1'_K(P)$.*
   (b) *Form t tables of size $m \times t$ as follows: For each table, randomly choose m distinct start points $X_{i,0}$, $i = 0, 1, \ldots, m-1$ and compute m chains of values of length t, $X_{i,1} = f(X_{i,0})$, $X_{i,2} = f(X_{i,1})$, $\ldots$, $X_{i,t} = f(X_{i,t-1})$, where $f(x) = h \circ g(x)$ and h is a simple reversible modification of the output of f (e.g. bit shuffling) if $n_1 = B'$; otherwise, h is a truncation from B' bits to n_1 bits followed by a simple transformation if $n_1 < B'$.*

This will form a table of size $m \times t$. Repeat this process to form t such tables where all the start points are distinct. Each h should be different for all t tables. We expect to cover most of the key space of K_1 by this process, which has complexity $mt^2 \cdot \lceil \frac{n_1}{B} \rceil = 2^{n_1} \cdot \lceil \frac{n_1}{B} \rceil$.

(c) To reduce memory requirements, discard all intermediate points and sort the start and end points $(X_{i,0}, X_{i,t})$ according to the end points $X_{i,t}$. Store the start and end points of each table using $mt \cdot \lceil \frac{n_1}{B} \rceil$ memory.

2. Attack:

(a) Compute $C' = h(E2'^{-1}_{K_2}(C))$ over all possible values of K_2.

(b) For a particular K_2, check to see if C' is equal to an end-point $X_{i,t}$ of a table. If it is, then we can guess that $(X_{i,t-1}, K_2)$ is a possible encryption key. The value $X_{i,t-1}$ can be computed from $f^{t-1}(X_{i,0})$. Check whether $E2'^{-1}_{K_2}(C) = E1'_{X_{i,t-1}}(P)$ to see if $(X_{i,t-1}, K_2)$ is a possible key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1}}(PT_i)) = CT_i$ for $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1}, K_2)$ if it does not satisfy this equation for any i.

(c) If not, compute $f^j(C')$, $j = 1, \ldots, t-1$ and check to see if it is equal to an end-point $X_{i,t}$ of a table. If it is, then $(X_{i,t-1-j}, K_2)$ is a possible encryption key. The value $X_{i,t-1-j}$ can be computed from $f^{t-1-j}(X_{i,0})$. Again, check whether $E2'^{-1}_{K_2}(C) = E1'_{X_{i,t-1-j}}(P)$ to see if $(X_{i,t-1-j}, K_2)$ is indeed a possible key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1-j}}(PT_i)) = CT_i$ for $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1-j}, K_2)$ if it does not satisfy this equation for any i. The complexity of covering a table (excluding verification) is $t \cdot \lceil \frac{n_1}{B} \rceil$.

(d) If the key (K_1, K_2) is not found in a table, then repeat steps 2(b) and 2(c) for the other $t-1$ tables to find the key. Thus the total complexity of covering these tables for all keys K_2 (excluding verification) is $t^2 \cdot 2^{n_2} \cdot \lceil \frac{n_1}{B} \rceil$.

Algorithm 5: Rainbow-MTM Attack

1. Pre-processing:

(a) Choose two positive integers m, t such that $2^{n_1} = mt^2$. Fix a plaintext P and define a one-way function $g(K) = E1'_K(P)$.

(b) Form a table of size $mt \times t$ as follows: Randomly choose mt distinct start points $X_{i,0}$, $i = 0, 1, \ldots, mt-1$ and compute mt chains of values of length t, $X_{i,1} = f_1(X_{i,0})$, $X_{i,2} = f_2(X_{i,1})$, $\ldots$, $X_{i,t} = f_t(X_{i,t-1})$, where $f_i(x) = h_i \circ g(x)$ and h_i is a simple reversible modification of the output of f (e.g. bit shuffling) if $n_1 = B'$; otherwise, h_i is a truncation from B' bits to n_1 bits followed by a simple transformation if $n_1 < B'$. This will form a table of size $mt \times t$. Ensure that all the mt endpoints are distinct. This pre-processing step has complexity $mt^2 \cdot \lceil \frac{n_1}{B} \rceil = 2^{n_1} \cdot \lceil \frac{n_1}{B} \rceil$.

(c) Sort the start and end points $(X_{i,0}, X_{i,t})$ according to the end points $X_{i,t}$. Store them using $mt \cdot \lceil \frac{n_1}{B} \rceil$ memory.

2. Attack:

(a) Compute $C' = E2'^{-1}_{K_2}(C)$ over all possible values of K_2.

(b) *For a particular K_2, check to see if $h_t(C')$ is equal to an end-point $X_{i,t}$ of a table. If it is, then we can guess that $(X_{i,t-1}, K_2)$ is a possible encryption key. The value $X_{i,t-1}$ can be computed by rebuilding the chain from the corresponding start point $X_{i,0}$. Check whether $C' = E1'_{X_{i,t-1}}(P)$ to see if $(X_{i,t-1}, K_2)$ is indeed a possible encryption key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1}}(PT_i)) = CT_i$ for $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1}, K_2)$ if it does not satisfy this equation for any i.*

(c) *If the correct key is not found, compute $f_t \circ f_{t-1} \circ \ldots \circ f_{t-j+1} \circ h_{t-j}(C')$, $j = 1, \ldots, t-1$ and check to see if it is equal to an end-point $X_{i,t}$. If it is, then $(X_{i,t-1-j}, K_2)$ is a possible encryption key. The value $X_{i,t-1-j}$ can be computed from the start point $X_{i,0}$. Again, check whether $C' = E1'_{X_{i,t-1-j}}(P)$ to see if $(X_{i,t-1-j}, K_2)$ is indeed a possible key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1-j}}(PT_i)) = CT_i$ for $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1-j}, K_2)$ if it does not satisfy this equation for any i.The total complexity of covering this table for all keys K_2 (excluding verification) is $\frac{t(t+1)}{2} \cdot 2^{n_2} \cdot \lceil \frac{n_1}{B} \rceil \approx \frac{t^2}{2} \cdot 2^{n_2} \cdot \lceil \frac{n_1}{B} \rceil$.*

Algorithm 6: BS-MTM Attack

1. *Pre-processing:*
   (a) *Choose two positive integers m, t such that $2^{n_1} = mt^2$. Fix a plaintext P and define a one-way function $g(K) = E1'_K(P)$.*
   (b) *Form $t/2^d$ tables of size $m \times t$ as in step 1(b) of Algorithm 4 with complexity $mt^2/2^d \cdot \lceil \frac{n_1}{B} \rceil = 2^{n_1-d} \cdot \lceil \frac{n_1}{B} \rceil$. This will cover $1/2^d$ of the keyspace of K_1.*
   (c) *Sort and store the start and end points of each table using $mt/2^d \cdot \lceil \frac{n_1}{B} \rceil$ memory as in step 1(c) of Algorithm 4.*
2. *Attack:*
   (a) *Compute $C'_0 = h(E2'^{-1}_{K_2}(C_0))$ over all possible values of K_2.*
   (b) *Search to see if $C'_0, f(C'_0), \ldots, f^{t-1}(C'_0)$ is equal to one of the end points of any of our stored tables and compute the key K_1 from the corresponding start point as in steps 2(b), 2(c) and 2(d) of Algorithm 4. The complexity of covering these tables for all keys K_2 (excluding verification) is $t/2^d \cdot t \cdot 2^{n_2} \cdot \lceil \frac{n_1}{B} \rceil$.*
   (c) *If the correct key $\left(K_1^{(0)}, K_2^{(0)}\right)$ is not contained in the space of 2^{n_1-d} keys computed, then proceed to repeat the attack for C_j, $j = 1, \ldots, 2^d - 1$ consecutively until one correct key is found. Note that the set of $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ plaintext-ciphertext pairs used for verification phase need not be uniform across all the different C_j.*

Algorithm 7: TMD-MTM Attack

1. *Pre-processing:*
   (a) *Fix a plaintext P and define a one-way function $g(K) = E1'_K(P)$.*

(b) Form r tables of size $m \times t$ as in step 1(b) of algorithm [4]. The parameters (r, m, t) are chosen such that the tables will cover $1/2^d$ of the keyspace of K_1 (where $D = 2^d$ is the number of encryptions of P using different keys). The exact conditions that they must satisfy will be given below.

(c) Sort and store the start and end points of each table using $rm \cdot \lceil \frac{n_1}{B} \rceil$ memory as in step 1(c) of Algorithm [4].

2. Attack:

(a) Compute $C_0' = h(E2'^{-1}_{K_2}(C_0))$ over all possible values of K_2.

(b) Search to see if $C_0', f(C_j'), \ldots, f^{t-1}(C_0')$ is equal to one of the end points of any of our stored tables and compute the key K_1 from the corresponding start point as in steps 2(b), 2(c) and 2(d) of Algorithm [4].

(c) If the correct key $\left(K_1^{(0)}, K_2^{(0)}\right)$ is not contained in the space of 2^{n_1-d} keys computed, then proceed to repeat the attack and verification for C_j, $j = 1, \ldots, 2^d - 1$ consecutively until one correct key is found. Note that the set of $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ plaintext-ciphertext pairs used for verification phase need not be uniform across all the different C_j.

In our TMD-MTM attack, we shall assume that the number of columns of each table is $\gg 1$ and

$$\text{time for one table look-up} \approx \text{time for one invocation of } f.$$

Then we have the relations:

$$\left.\begin{array}{rl}
N & = 2^{n_1} \\
PC & = rmt \quad (\#\ g \text{ invocations in the pre-computation phase}) \\
 & = \frac{N}{D} \quad (\text{coverage}) \\
PC' & = PC \cdot \lceil \frac{n_1}{B} \rceil \quad (\text{pre-computation complexity}) \\
M & = rm \\
M' & = M \cdot \lceil \frac{n_1}{B} \rceil \quad (\text{memory}) \\
D & = 2^d \quad (\#\ \text{encryptions of } P \text{ using different keys}) \\
T & = rtD \\
T' & = 2^{n_2} \cdot T \cdot \lceil \frac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right) \\
 & \approx 2^{n_2+d} rt + 2^{n_2+d} \quad (\text{time for online phase}) \\
mt^2 & \leq N \quad (\text{birthday bound}) \\
T' & < PC'
\end{array}\right\} \qquad (1)$$

The last inequality in (1) was added since in practical attacks, we usually require the online attacking time to be less than the offline table preparation time. We can solve for r, m and t to get:

$$\left.\begin{array}{rll}
t & = \frac{N}{MD} \geq 1 & (\text{number of columns}) \\
m & = \frac{N}{T} & (\text{number of rows}) \\
r & = \frac{MT}{N} \geq 1 & (\text{number of tables}) \\
mt^2 & = \frac{N^3}{TM^2D^2} \leq N & (\text{birthday bound})
\end{array}\right\} \qquad (2)$$

Based on these, we can derive the TMTO curve given by:

$$\left.\begin{aligned} D &= N^w \\ MT &= N^x \\ M &= N^y \\ mt^2 &= N^z \\ PC &= N^{1-w} \\ T &= N^{x-y} \end{aligned}\right\} \tag{3}$$

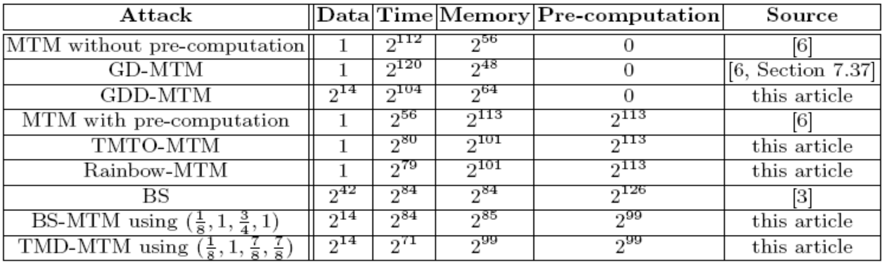
where

$$\left.\begin{gathered} 2w + x + y + z = 3 \\ 0 \le w < 1 \\ 0 \le y, x - y < 1 \le x \\ w + y \le 1 \\ 0 \le z \le 1 \\ \tfrac{n_2}{n_1} < 1 - (w + x) + y \end{gathered}\right\} \tag{4}$$

Therefore, any set of parameters (w, x, y, z) satisfying (4) gives a valid attack. We can also express (r, m, t) in terms of (w, x, y, z) as follows:

$$\left.\begin{aligned} r &= N^{x-1} \\ m &= N^{1-(x-y)} \\ t &= N^{1-w-y} \end{aligned}\right\} \tag{5}$$

B Tables



Attack	Data	Time	Memory	Pre-computation	Source
MTM without pre-computation	1	2^{112}	2^{56}	0	[6]
GD-MTM	1	2^{120}	2^{48}	0	[6, Section 7.37]
GDD-MTM	2^{14}	2^{104}	2^{64}	0	this article
MTM with pre-computation	1	2^{56}	2^{113}	2^{113}	[6]
TMTO-MTM	1	2^{80}	2^{101}	2^{113}	this article
Rainbow-MTM	1	2^{79}	2^{101}	2^{113}	this article
BS	2^{42}	2^{84}	2^{84}	2^{126}	[3]
BS-MTM using $(\frac{1}{8}, 1, \frac{3}{4}, 1)$	2^{14}	2^{84}	2^{85}	2^{99}	this article
TMD-MTM using $(\frac{1}{8}, 1, \frac{7}{8}, \frac{7}{8})$	2^{14}	2^{71}	2^{99}	2^{99}	this article

Fig. 1. Attacks on 3-key triple DES

Beyond User-to-User Access Control for Online Social Networks

Mohamed Shehab[1], Anna Cinzia Squicciarini[2], and Gail-Joon Ahn[3]

[1] University of North Carolina at Charlotte, NC, USA
[2] The Pennsylvania State University, PA, USA
[3] Arizona State University, AZ, USA
mshehab@uncc.edu, acs20@psu.edu, gahn@asu.edu

Abstract. With the development of Web 2.0 technologies, online social networks are able to provide open platforms to enable the seamless sharing of profile data to enable public developers to interface and extend the social network services as applications (or APIs). At the same time, these open interfaces pose serious privacy concerns as third party applications are usually given full read access to the user profiles. Current related research has focused on mainly user-to-user interactions in social networks, and seems to ignore the third party applications. In this paper, we present an access control framework to manage the third party to user interactions. Our framework is based on enabling the user to specify the data attributes to be shared with the application and at the same time be able to specify the degree of specificity of the shared attributes. We model applications as finite state machines, and use the required user profile attributes as conditions governing the application execution. We formulate the minimal attribute generalization problem and we propose a solution that maps the problem to the shortest path problem to find the minimum set of attribute generalization required to access the application services.

1 Introduction

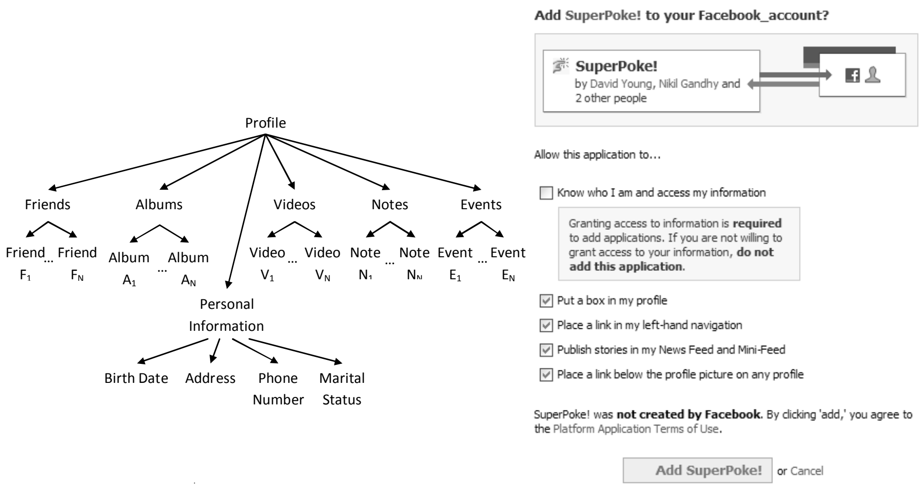
The recent growth of social network sites such as Facebook, del.icio.us and Myspace have created many interesting and challenging problems to the research communities. In social networks users self-organize into different communities, and manage their own profile, as a form of self-expression. Users profiles usually include information such as the user's name, birthdate, address, contact information, emails, education, interests, photos, music, videos, blogs and many other attributes. The structure of an example social network profile is depicted in Figure 1(a). Controlling access to the user profile information is a challenging task as it requires average internet users to act as system administrators to specify and configure access control policies for their profiles. To control interactions between users, the user's world is divided into a trusted and a non-trusted set of users, referred to as *friends* and *strangers* respectively. Furthermore, some social networks allow users to further partition the set of friends by geographical

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 174–189, 2008.

location, social group, organization, or by how well they know them. Users are provided with group based access control mechanisms that apply access rules on the different groups of friends and strangers. Facebook [6] enables users to create a limited profile and to select which users map to that profile. For example, a user could share his wedding album with his family members and not with his colleagues from work. In addition to the issues involved with enabling fine grain access control for each user profile [5] to control data attributes viewable by other users, a yet unexplored problem is related to users' profile access from entities different from other social network users.

With the development of Web 2.0 technologies [20], online social networks are able to provide open platforms to enable the seamless sharing of profile data to enable public developers to interface and extend the social network services as applications (or APIs). For example, Facebook allows anyone to create software plug-ins that can be added to user profiles to provide services based on profile data. These features have been a great success, the most popular Facebook applications have around 24 million users as of May 2008, and competing social networking sites have moved to create their own imitation platforms. However, although these open platforms enable such advanced features, they also pose serious privacy risks. Users' profiles in fact have a great commercial value to marketing companies, competing networking sites, and identity thieves. Data mining through the development platform can potentially affect more people than screen scraping, because it exposes information that might otherwise be hidden.

Applications that are currently added to the users' profiles are given full read access to all the profiles information [6,18]. The user is able to add the



Add SuperPoke! to your Facebook_account?

SuperPoke!
by David Young, Nikil Gandhy and 2 other people

Allow this application to...

☐ Know who I am and access my information

Granting access to information is **required** to add applications. If you are not willing to grant access to your information, **do not add this application**.

☑ Put a box in my profile

☑ Place a link in my left-hand navigation

☑ Publish stories in my News Feed and Mini-Feed

☑ Place a link below the profile picture on any profile

SuperPoke! was **not created by Facebook**. By clicking 'add,' you agree to the Platform Application Terms of Use.

(a) Example User Profile Schema (b) App. Addition Error Message

Fig. 1. Social Networks Profiles and Applications

application only if he/she agrees to give the application access not only to his profile, but also to profile data of other users viewable through that user. In other words, the user enables the application to read information on his behalf. If the user refuses to grant full read access to the application the installation process fails. For example, Figure 1(b), shows the error message displayed by the Facebook platform when the user rejects to give the application full read access to his profile data. Basically, the application access control model adopted by the request management module is an *all-or-nothing* policy. As such, API developers have access to users' data, regardless of the actual applications' needs, leading to potentially serious privacy breaches. Such information flow is often hidden or not clear to social network users, who are often not aware of the amount of data that is actually being disclosed, since they do not really distinguish among social network users and developers outside the social network boundaries. We believe, in order to promote healthy development of social networks environments and protect fundamental individuals' privacy rights, users should be in control at any time of their data and be well informed about their usage. Applications should be given limited privileges to the user profile and only given access to the smallest set of profile data required to perform their tasks. For example, a horoscope application should be given access to only the birthday information, while a fortune cookie application that displays a random daily quote on the user's profile should not be given access to any profile data.

Although this issue has been recognized by the media [16,2,4] and by social network users, to date, no technical solution has been proposed so far. Ideally, users' should be able to take advantage of the available applications while still having a stronger control on their data. The problem is not trivial, in that it requires new access control models for APIs in social networks, as well as extending social network applications. Applications should be designed so to be customized, based on users' profile preferences and second, users should have the ability to specify the data that they are willing to reveal. Additionally, users should be able to use data privacy mechanisms such as generalization to enjoy the services provided through APIs without having to disclose identifying or private information.

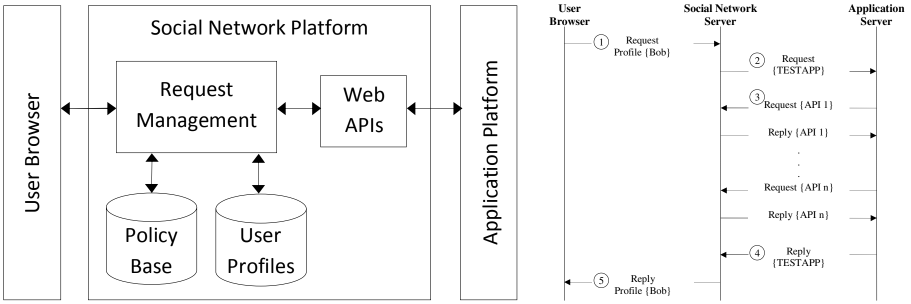
In this paper we address this issue, by deploying an access control mechanism for applications in social networks. Our goal is to provide a privacy-enabled solution that is in line with social network ethics of openness, and does not hinder users' opportunities of adding useful and entertaining applications to their profiles. Our access control mechanism is based on enabling the user to specify the data attributes to be shared with the application and at the same time be able to specify the degree of specificity of the shared attributes. Enabling such a mechanism requires application to be developed to accommodate different user preferences. We model applications as finite state machines, and use the required user profile attributes as conditions governing the application execution. The challenge the user is faced with is what is the minimum set of attributes and their minimum generalization levels required to acquire specific services provided by the application. In order to address this problem we proposed the weighted application transition system and formulated the Minimal Attribute Generalization Problem. Furthermore, we propose a solution that maps the problem to the shortest path problem to find the minimum set of attribute generalization required to access the application services.

The rest of the paper is organized as follows. In Section 2, we provide background information related to Social Network APIs. In Section 3, we introduce our developer APIs access control framework. In Section 4, we discuss how to provide customized applications. Section 5 describes the related work. The conclusion and future work are discussed in Section 6.

2 Background on Social Network APIs

With the emergence of new web technologies, and with the establishment of the Web 2.0, a large number of web sites are exposing their services by providing web programming interfaces (APIs). For example, Google Web API [12] provides a programming interface to query web pages through Google from user developed applications. Several social network web sites have released APIs that allow developers to leverage and aggregate information stored in user profiles and provide extended social network services. The exposed APIs are basically a set of web services that provide a limited and controlled view for the application to interface with the social network site. The social network application architecture includes three interacting parties namely the user, social network server, and the third party application server. Figure 2(a), shows the different blocks used in a the social networks architecture. Note that, the application server is able to connect to social network through the exported web APIs. Furthermore, these requests are filtered through the request management module which will be discussed in detail in the next section.

For example, consider an application that recommends stores in your area that are having sales. In this case, the application requires to retrieve your address, age, marital status, and gender. The address information is required to be able



(a) Social Network System Architecture (b) Application Interactions

Fig. 2. Social Network Architecture and Application Interactions

to locate shops in your region, and the other parameters are required to provide a more focused recommendation. Some other applications would not only require data from your profile but would also require data from your friends' profiles. For example, consider an application that projects your friends on an online map according to the address listed on their profiles. This application requires your address and your friend list, then for each friend it would retrieve their address.

Social networks provide mechanisms for users to customize their profiles and to add applications developed by external developers. The application provides the customized services by accessing the exported APIs. Figure 2(b), depicts the interaction stages between the user browser, social network and the third party developer application. The interaction starts when a user requests an application *APP* (Steps 1-2). The application server interacts with the social network server by instantiating API calls (Step 3). Upon receiving the responses of the API calls, the application server compiles and sends a response to the social network which is forwarded to the requesting user (Steps 4-5). Note that in this model the social network outsources the application development and execution to an external third party application server.

3 Developer APIs Access Control Framework

Applications require to access user's profile data to provide a service customized to the user's profile data. In this section we present our approach to enable fine grain access control [5,21] for developer's applications, to limit applications' access only to relevant user's profile data. We first provide some preliminary definitions related to applications and API set, then discuss our proposed fine grain access control framework for API based applications, and then focus on the relevant phases that characterize our approach.

3.1 Social Network Profiles and Data Sets

For the purpose of our work, the two main components of a social network are represented by applications and users' profiles.

Users' Profiles. Users' profiles are modeled as collection of data items that are uniquely associated to them. Each data item is defined over a finite domain of legal values.

Definition 1. *(User Profile) A user profile for user i, is characterized by an attribute vector* $x = \{x_1, \ldots, x_n\}$*, where attribute* x_i *takes values in a domain* D^i*, which also includes the null value referred to by* $\perp$*.*

Profile data items in our approach can be generalized to increase privacy of users. A common practice in privacy preservation mechanisms is to replace data records with suppressed or more general values [24,25] in order to ensure anonymity and prevent disclosure of sensitive data. A simple disclosure policy can simply suppress an attribute if certain disclosure criterion are met, in this case that is a all or none policy. A generalization disclosure policy, is accomplished by assigning a disclosed value that is more general than the original attribute value. For example, the user can make the address information less specific by omitting the street and city and revealing just the zip code. Figure 3, shows an example partial value generalization hierarchy of the address attribute. We assume that domain D^i for a certain data item x_i (see Definition 1) is a partially ordered set $(D^i_j, \prec)$, where D^i_j are the attribute generalizations and $\prec$ is the ordering operator. In the domain D^i the largest element corresponds to the non-generalized attribute value and the smallest element is the most generalized value which is the suppressed value $\perp$. The domain D^i contains l_i generalization levels, an attribute generalized to the h^{th} level of generalization is denoted by D^i_h, where $0 \leq h < l_i$. Data attribute generalized to D^i_1 is more general than an attribute generalized to D^i_2, $D^i_2 \prec D^i_1$, which implies that D^i_2 discloses more information than D^i_1. Given a user profile x, by specifying generalization preferences for each of the profile attributes the user is able to specify a different view for each application. The user generalization preferences for an application is defined by the attribute generalization vector $UP = [h_1, \ldots, h_n]$ where h_i represents the generalization level D^i_h permitted for profile attribute i. Different attributes have different disclosure sensitivity, for example some users might regard their home address more sensitive than their cell phone number. To capture attribute sensitivity, for each profile attribute $x_i \in x$ the user assigns a *sensitivity metric* Φ_i, which is specified for the non-generalized attribute $D^i_{l_i-1}$. Note that the sensitivity of an attribute x_i generalized to level h_i is proportional to $\Phi_i h_i$. Given a user generalization preference vector the $UP = [h_1, \ldots, h_n]$, the risk of attribute disclosure is proportional to $\Theta(UP) = \sum_{i=1}^{n} \Phi_i h_i$. Note that the function $\Theta()$ provides a mechanism to compare user generalization preferences. The generalization model can be applied not only to the data explicitly mentioned on the profile in addition it can be applied to the tags and the metadata that are attached to the profile data.

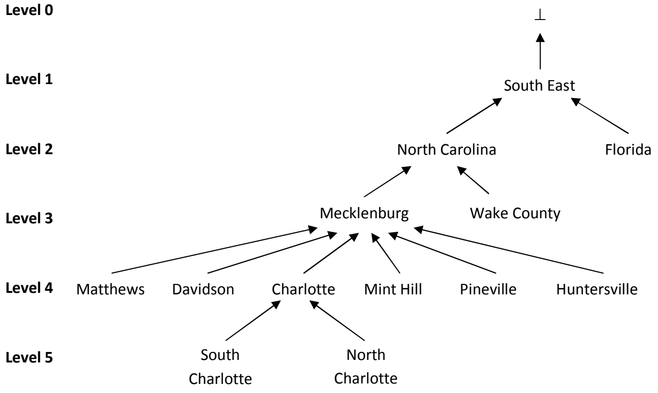


Fig. 3. A partial value generalization hierarchy of the address field

Applications. The building block for our model is represented by applications. Applications are composed of a set of API's which are functions called by the application.

Definition 2. *(Application API Set). Given an application App, the application API set App.apiset is the set of APIs called by application App, represented as the set* $App.apiset = \{api_1, \ldots, api_n\}$.

For example, consider a horoscope application *HoroAPP*, illustrated in Figure 4. It calls the "*user.get_birthday*()" and "*user.get_friends*()" APIs. The application API set for *HoroAPP* is *HoroAPP.apiset* = {*user.get_birthday*(), *user.get_friends*()}. From the API calls the set of data set accessed by the application can be obtained by tracing the data acquired by the called API's. For example, consider an API "*user.get_birthday*()", the profile data accessed is {*profile.birthday*}. Other APIs involve the processing of several profile data items for example, consider the API "*user.get_photos_with_friends*()", this API returns the photos taken with friends. The API performs a join between the user friends and the user photo album meta data, in this case the data items access are {*profile.ablums*, *profile.friends*}. Accordingly, an application can be translated from a set of API calls to a set of data accesses. This set of accessed data can be then presented to the user to enable the selection of what data items to expose.

```
ExampleApplication(){
  a = get_friends(userid);
  ...
  b = get_albums(userid);
  ...
  Query = "SELECT birthday FROM user_db
                WHERE uid=userid";
  c = send_query(Query);
  ...
}
```

Fig. 4. Example of horoscope application

3.2 The Access Control Framework

Our framework adopts the *Principle of Least Privilege* [23], which requires that each principal be accorded the minimum access privileges needed to accomplish its task. In our context, principals are the application developers, and the application should be awarded access to the minimum set of profile data in order to provide the requested service. To achieve this goal we present a mechanism that enables fine grain access control on the profile data. Such a mechanism enables the application developer to select the data items required by the application and at the same time enables the user to opt-in or opt-out or generalize each of the requested data items. Specifically, our framework is characterized by three main phases: *application registration*, to register the application at the social network server; *user application addition*, to add the application in a local profile; and *application adaptation*, within which the application adapts according to the provided data items. We discuss them in what follows.

Application Registration. The application developers register the application with the social network server. The developers are required to share the application API calls and the application business state diagram describing the application process, the details of this requirement will be discussed in following sections. As part of the registration process, developers need to tag the application, by labeling each API within the application with the set of user's data items used by the application. The tags provided during this stage only refer to the user's profile data involved and do not include any external output or additional user input that may be required when executing the API. The provided application information is used to compile an *application sheet* describing the data attributes required by the application.

User Application Addition. Once the application is registered with the social network server, it becomes available for social network users to add to their social network profiles. Upon selecting the application, the application sheet is presented to the user, who is prompted with the following options for each data item required by the API: choose to opt-in, opt-out, or generalize. Intuitively, the user opts-in for the data items he is willing to disclose to the application. If the user opts-out for some data the application needs to adapt in order to be properly executed without such input. In case the generalize option is chosen for certain data item, then the user only accepts the application to employ generalized data attribute [24,25]. The user selections are input in the *user sheet*, which indicates the user access preference for the added application.

```
<APPSHEET>
  <APP id="332198764">
    <DESCRIPTION>
      <NAME> Horoscope App </NAME>
      <INFO> Provide daily horoscope
       from www.horoscope.com </INFO>
    </DESCRIPTION>
    <DATA-GROUP>
      <DATA ref="profile.birthday"/>
      <DATA ref="profile.gender"/>
      <DATA ref="profile.address"/>
    </DATA-GROUP>
  </APP>
</APPSHEET>
```

(a) Application Sheet

```
<USERSHEET>
  <APP id="332198764">
    <ALLOW>
      <DATA-GROUP>
        <DATA ref="profile.birthday.day"/>
        <DATA ref="profile.birthday.month"/>
      </DATA-GROUP>
    </ALLOW>
  </APP>
</USERSHEET>
```

(b) User Sheet

Fig. 5. Application and User Sheets

An example of XML encoding for the horoscope application is reported in Figure 5. In Figure 5(a) we report the application sheet, where birthday, gender and address are requested. In Figure 5(b) we report the user sheet in case the user opted to disclose only month and year of birth.

User Application Adaptation. At this stage the user sheet is used to generate a version of the application executable using the input obtained by the profile data items. This phase requires the application to differentiate provisioning according to the permissible data items and their respective generalization levels. We discuss in the next section how this not trivial task is achieved.

4 Customized Application Service Provisioning

The user sheet provides a mechanism for users to specify generalization preferences on the profile attributes to restrict the data accessible to the application. On the other hand, by enabling attribute generalizations the application is faced with the problem of missing data, and might not ensure the provisioning of the request service based on the provided data generalizations. To address this issue we propose that during the application registration phase the application developer is required to provide the process execution description of the application. The process execution description describes the interactions between the composed APIs. A candidate process description language standard is BPEL (Business Process Execution Language for Web Services, also WS-BPEL, BPEL4WS) [19] which provides a rich vocabulary for expressing composition, orchestration and coordination of web services to describe the behavior of business processes. Figure 6, shows an example process execution diagram describing the service invocations and service transitions required by an application that aggregates the user's friends' addresses and projects them on Google Maps. Note that the transitions are labeled with conditions on the returned API calls. The web services composition and choreography described by BPEL can be formalized based using finite state processes (FSP) [8,22,7]. In what follows we define the application as a transition system.

Definition 3. *(Application Transition System). An application transition system is a tuple* $TS = (S, \Sigma, \delta)$*, where:*

- S *is a finite set of states. The set of states includes a single initial state* s_0 *and a finite set of final states* $F \subseteq S$*.*
- Σ *is the alphabet of operations offered by the service and the data required by this service.*
- $\delta : S \times \Sigma \rightarrow S$ *is the transition function that maps states and alphabets to another state. The transition* $\delta(s_i, \alpha) = s_j$*, represents that transition from state* s_i *to state* s_j *subject to services and data in* α*.*

The mapping function δ is used to represent the constraints required to transition from one state to another. In this paper, we focus on constraints related to the required profile data generalization levels requested by the application to enable the successful transition from a state to another. For example, an application requesting the user's address through the service *get_address*(), the application will transition to a different state depending on the generalization level of the returned address attribute. From an application perspective the user generalization preference vector specifies the permitted attribute generalization levels, which in turn dictates the set of permissible state transitions. The set of final states represents the different service levels provided by the application.

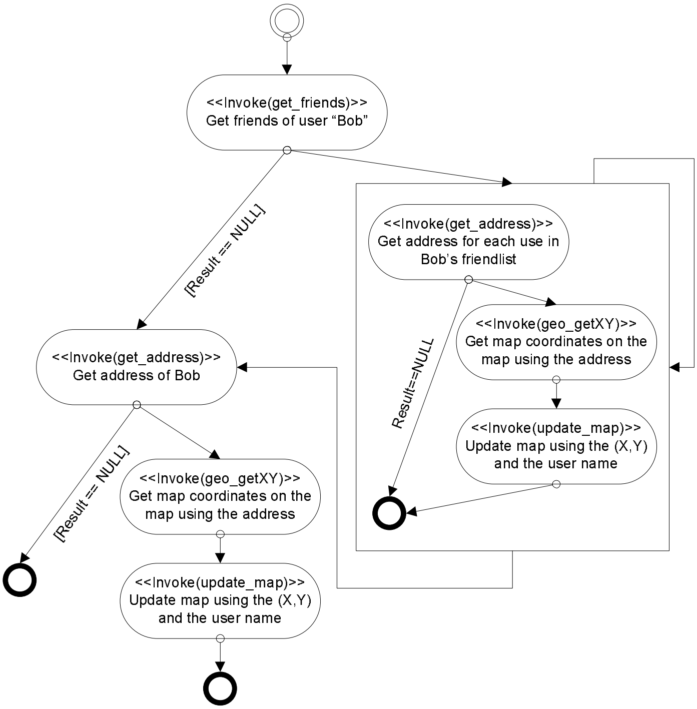

<<Invoke(get_friends)>>
Get friends of user "Bob"
[Result == NULL]
<<Invoke(get_address)>>
Get address for each use in Bob's friendlist
Result==NULL
<<Invoke(geo_getXY)>>
Get map coordinates on the map using the address
<<Invoke(update_map)>>
Update map using the (X,Y) and the user name
<<Invoke(get_address)>>
Get address of Bob
[Result == NULL]
<<Invoke(geo_getXY)>>
Get map coordinates on the map using the address
<<Invoke(update_map)>>
Update map using the (X,Y) and the user name

Fig. 6. Example Application Process

Definition 4. *Given an application transition system $TS = (S, \Sigma, \delta)$ and a user preference vector UP, the reduced application transition system TS_{UP} is defined as the tuple $(S_R, \Sigma_R, \delta_R)$, where:*

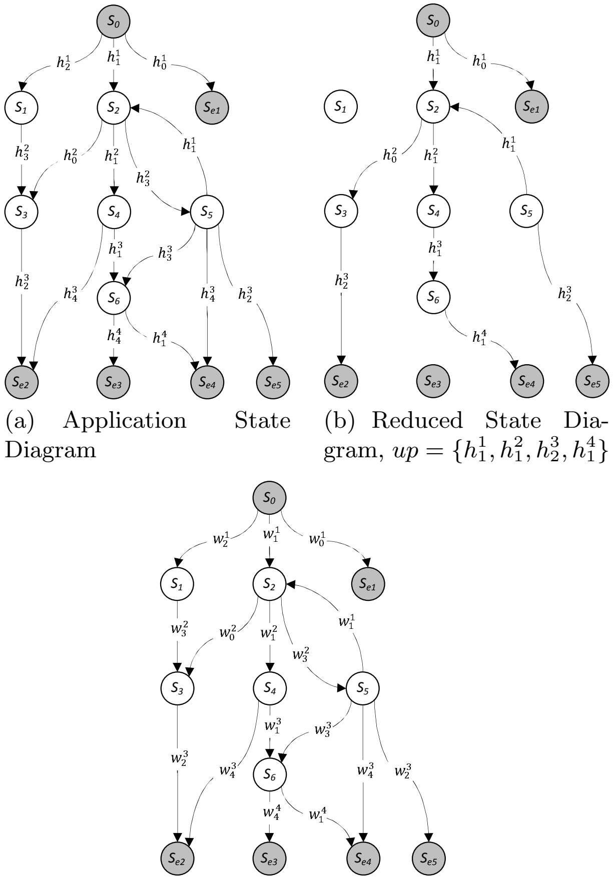
- *$S_R = S$ and $\Sigma_R = \Sigma$.*
- *$\delta_R = \delta$ for $\delta(s_i, \alpha) = s_j$ where the attributes α satisfies the user preference vector UP.*

The reduced application transition system includes only the state transitions that are permitted by the user preferences. It also indicates the states that are reachable after the user preferences are applied to the application.

We model the application transition system TS as a directed graph $G = (V, E)$, where the vertices V represent the states, and the edges E represent the state transitions. The edges E are labeled with the minimum attribute generalization levels required to enable the state transition. For an edge $e \in E$ the edge label $e.h$ represents the generalization level required for the state transition. For example, in Figure 7(a) the edge (S_0, S_1) is labeled with h_2^1 indicating that the generalization level 2 is required for attribute x_1 to enable transition from state S_0 to state S_1. A user preference is said to satisfy a transition if the specified user attribute generalization level is greater than or equal to the edge generalization level. The reduced application transition system is computed by generating a graph $G_R = (V_R, E_R)$, where $V_R = V$ and $E_R \subseteq E$ includes only the transitions E that satisfy the user preferences. Figure 7(b), shows an example reduced application transition graph for the user preference vector $up = \{h_1^1, h_1^2, h_2^3, h_1^4\}$ and the original application state diagram in Figure 7(a).

Definition 5. *(Application Service Path) Given an application transition instance TS, the path $P = \{e_0, \ldots, e_{n-1}\}$ is sequence of state transitions, where the edge e_0 starts at the initial state s_0 and the ending edge e_{n-1} terminating at a final state $s_n \in F$. The path generalization vector $g(P) = \{e_1.h, \ldots, e_{n-1}.h\}$ is defined as the set of data attribute generalization levels required to traverse this path.*

The Application Service Path represents an instance of an application execution that starts at the start state s_0 and ends at a target ending state s_n.



(a) Application State Diagram

(b) Reduced State Diagram, $up = \{h_1^1, h_1^2, h_2^3, h_1^4\}$

(c) Weighted State Diagram, $w_j^i = h_j^i * \Phi^i$

Fig. 7. Application State Diagram and User Preferences

4.1 Optimal User Application Preferences

In our framework, when trying to install an application, the user specifies an attribute generalization preferences and a target final application state. The challenge the user is faced with is to identify the minimal attribute generalization preference required to enable the application to successfully terminate to the requested final state. According to the well-known security principle of *Least Privilege* [23], which requires that each principal be accorded the minimum access privileges needed to accomplish its task, this translates to the requirement that an application should be awarded the access to the smallest set of profile attributes at the minimum generalization levels in order to provide the requested service. Formally, the minimal attribute generalization problem is defined as follows:

Definition 6. Minimal Attribute Generalization Problem*, Given an application transition instance $TS = (S, \Sigma, \delta)$, and a target final state $s_f \in F$, determine the minimal user attribute vector $UP^* = [h_1^*, \ldots, h_n^*]$ required to enable the successful transition from the start state s_0 to the final state s_f.*

The minimal user attribute vector is the vector that requires the minimum exposure of the user attributes and enables the application to transition to the target final state. Using the graph based application transition model, an application service path beginning at start state and terminating at the final target state holds the set of generalization levels required to take such a path. The minimal attribute generalization problem translates to finding the minimal application service path from the start state to the target final state in a *weighted application transition system* defined as follows:

Definition 7. *(Weighted Application Transition System). A weighted application transition system $TSW = (G, W)$ where:*

- *G is the application transition graph $G = (V, E)$, where V is the set of vertices representing the finite set of states, and E is the set of edges representing the state transitions.*
- *$W : E \times \Phi \rightarrow w \in \Re^+$ is the edge weight function that maps the edge attribute generalization labeling $E.h$ and the attribute sensitivity Φ to an edge weight w.*

Given an application service path $P = \{e_0, \ldots, e_{n-1}\}$, the path length is defined as follows:

$$\Theta(UP) = \sum_{i=0}^{n-1} W(e_i, \Phi_i) = \sum_{i=0}^{n-1} \Phi_i e_i.h$$

Given the weighted application transition system and the path length definition, the minimal attribute generalization problem simply maps to finding the shortest path from the start state s_0 to the final target state s_f. The initially specified user preferences are used as an upper limit on the user preferences and are referred to as the upper limit user preferences $UPL = [h_0, \ldots, h_n]$. Figure 8, depicts the algorithm used to compute the minimal user attribute preferences vector. Lines 1-9, initialize the application transition graph to generate the edges that are not allowed by the specified user attribute generalization upper limits buy setting the edge weights to ∞, and the weights of the permitted transitions using the edge weight function that incorporates both the user attribute sensitivity and generalization level. Lines 10-14, initialize the distance from s_0 to other vertices, where $d[u]$ and $pi[u]$ represent the shortest distance from s_0 to u and the predecessor of u on the shortest path respectively. Lines 15-24, computes the shortest path from s_0 to all the transition states. Lines 25-34, computes the minimal user preferences vector required to transition from state s_0 to the target final state s_f.

```
Algorithm: $generate\_minimal\_preference$
Input: Application transition graph $G = (V, E)$,
User upperlimit preferences $UPL = [h_0, \ldots, h_n]$, User target state $s_t$
Output: User Minimal Attribute Preferences $UP^*$

1.  $V_R \leftarrow V$
2.  $E_R \leftarrow E$
3.  //Generating the reduced graph
4.  for each $e \in E_R$
5.    for each $h \in UPL$
6.      if $h \prec e.h$
7.          $e.w = \infty$
8.      else
9.          $e.w = \Phi_{e.a} * e.h$
10. //Initialize distance from $s_0$
11. for each $v \in V_R$
12.   $d[v] = \infty$
13.   $pi[v] = \{\}$
14. $d[s_0] = 0$
15. //Computing Shortest Path from $s_0$
16. $S = \{\}$
17. $Q \leftarrow V_R$ //Priority Queue on $d[u]$
18. while $Q$ is not Empty
19.     $u = ExtractMin(Q)$
20.     $S \leftarrow S \cup \{u\}$
21.   for each $v \in adjacent(u)$
22.     if $d[v] > d[u] + w(u, v)$
23.         $d[v] = d[u] + w[u, v]$
24.         $pi[v] = u$
25. //Tracing Minimal User Preferences from $s_0$ to $s_t$
26. $UP^* = \{\}$
27. if $d[s_t] == \infty$
28.   return $UP^*$
29. $u = s_t$
30. do
31.     $UP^* = (pi[u], u).h \cup UP^*$
32.     $u = pi[u]$
33.  while $pi[u] \neq s_0$
34. return $UP^*$
```

Fig. 8. User Minimal Attribute Preferences Algorithm

5 Related Work

Security and privacy in Social Networks, and more generally in Web 2.0 are emerging as important and crucial research topics [15,1,14,10]. Several pilot studies conducted in the past few years have identified the need for solutions to address the problem of information leakage networks, based on interpersonal relationships and very flexible social interactions. Some social networking sites, such as FaceBook (http://www.facebook.com), have started to develop some forms of control, however the level of assurance are still limited. For example, FaceBook allows a user to join various networks (e.g., home university, home city) and control what information is released to each network. Further, a user can specify if a particular person should be "limited" from seeing particular material or blocked entirely from seeing any material. However, there is limited control over the amount of data API's can access related to user's data.

An interesting research proposal has been presented in [11], where a social-networking based access control scheme suitable for online sharing is presented. In the proposed approach authors consider identities as key pairs, and social relationship on the basis of social attestations. Access control lists are employed to define the access lists of users. A more sophisticated model has been proposed in [3]. The authors presented a rule-based access control mechanism for social networks. Such an approach is based on enforcement of complex policies expressed as constraints on the type, depth, and trust level of existing relationships. The authors also propose using certificates for granting relationships authenticity, and the client-side enforcement of access control according to a rule-based approach, where a subject requesting to access an object must demonstrate that it has the rights of doing that. However, both in both projects [11,3], the authors do not consider the issue of API's in their models, and they do not propose a method to control API's access to profile's personal data.

An ongoing research project is represented by PLOG [13]. The goal of PLOG is to facilitate access control that is automatic, expressive and convenient. The authors are interested in exploring content based access control to be applied in SN sites. We believe this is an interesting direction that we plan on investigating as extension of Private Box. Another interesting work related to ours is [9]. The authors present an integrated approach for content sharing supporting a lightweight access control mechanism. HomeViews facilitates ad hoc, peer-to-peer sharing of data between unmanaged home computers. Sharing and protection are accomplished without centralized management, global accounts, user authentication, or coordination of any kind. This contribution, although very promising does not specifically focus on SNs and thus the proposed solution, although inline with our selective approach to user's data is complementary to ours.

Some related work has also been conducted with specific focus on trust relationships in social networks. An important contribution on this topic has been proposed by [10]. The work introduces a definition of trust suitable for use in web-based social networks with a discussion of the properties that will influence its use in computation. The authors designed an approach for inferring trust relationships between individuals that are not directly connected in the network.

Specifically, they present TrustMail, a prototype email client that uses variations on these algorithms to score email messages in the user's inbox based on the user's participation and ratings in a trust network.

Our idea of transitional states was partly inspired by [17]. The authors propose a conversation-based access control model that enables service providers to retain some control on the disclosure of their access control policies while giving clients some guarantees on the termination of their interactions. Similarly to ours, the authors represent web service possible conversations as finite transition systems, in which final states in this context. Many have identified [14] the need of a new access control paradigm specific for so represent those in which the interaction with the client can be (but not necessarily) ended. We adopt a similar approach in that we represent possible applications as state machines, and we provide a labeling technique to enable the comparison of the possible application paths.

6 Conclusions

In this paper we have presented an access control framework for social networks developer applications that enables users to specify profile attribute preferences and requires applications to be designed so to be customized based on users' profile preferences. Our framework provided a privacy-enabled solution that is in line with social network ethics of openness, and does not hinder users' opportunities of adding useful and entertaining applications to their profiles. We modeled the applications as finite state machine with transition labeling indicating the generalization level required to enable application state transitions. We defined the reduced application transition system that only includes the state transitions possible with a given user generalization vector. Then we incorporated the user sensitivity metric to generate the weighted applications transition system.

Furthermore, we formalized the Minimal Attribute Generalization Problem and presented the Weighted Application Transition System which incorporates the user attribute sensitivity metric to generated a weighted graph representing the application state transitions. Using the weighted graph we transformed the Minimal Attribute Generalization Problem to the shortest path problem and provided an algorithm that generates the optimal user generalizations vector that will enable the transition to a target final state.

References

1. Acquisti, A., Gross, R.: Imagined communities: Awareness, information sharing, and privacy on the facebook. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 36–58. Springer, Heidelberg (2006)
2. CNET Blog. Exclusive: The next facebook privacy scandal (2008), http://news.cnet.com/8301-13739_3-9854409-46.html
3. Carminati, B., Ferrari, E., Perego, A.: Rule-based access control for social networks. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops (2). LNCS, vol. 4278, pp. 1734–1744. Springer, Heidelberg (2006)

4. Wahington Chronicle. Study raises new privacy concerns about facebook (2008), http://chronicle.com/free/2008/02/1489n.htm
5. Damiani, E., Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for XML documents. ACM Transactions on Information and System Security 5(2), 169–202 (2002)
6. Facebook (2007), http://www.facebook.com
7. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Ltsa-ws: A tool for model-based verification of web service compositions and choreography, pp. 771–774 (May 2006)
8. Foster, H., Uchitel, S., Magee, J., Kramer, J., Hu, M.: Using a rigorous approach for engineering web service compositions: a case study, vol. 1, pp. 217–224 (July 2005)
9. Geambasu, R., Balazinska, M., Gribble, S.D., Levy, H.M.: Homeviews: peer-to-peer middleware for personal data sharing applications. In: SIGMOD Conference, pp. 235–246 (2007)
10. Golbeck, J., Hendler, J.A.: Inferring binary trust relationships in web-based social networks. ACM Trans. Internet Techn. 6(4), 497–529 (2006)
11. Gollu, K.K., Saroiu, S., Wolman, A.: A social networking-based access control scheme for personal content. In: Proc. 21st ACM Symposium on Operating Systems Principles (SOSP 2007) (2007); Work in progress
12. Google Code. Google's Developer Network, http://code.google.com/
13. Hart, M., Johnson, R., Stent, A.: More content - less control: Access control in the Web 2.0. Web 2.0 Security & Privacy (2003)
14. Hogben, G.: Security issues and recommendations for online social networks. ENISA Position Paper N.1 (2007)
15. IEEE. W2SP 2008: Web 2.0 Security and Privacy (2008)
16. Irvine, M.: Social networking applications can pose security risks. Associated Press (April 2008)
17. Mecella, M., Ouzzani, M., Paci, F., Bertino, E.: Access control enforcement for conversation-based web services. In: WWW Conference, pp. 257–266 (2006)
18. MySpace (2007), http://www.myspace.com
19. OASIS. OASIS WSBPEL TC Webpage, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
20. O'Reilly, T.: What Is Web 2.0. O'Reilly Network, pp. 169–202 (September 2005)
21. Rizvi, S., Mendelzon, A., Sudarshan, S., Roy, P.: Extending query rewriting techniques for fine-grained access control. In: SIGMOD 2004: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pp. 551–562. ACM, New York (2004)
22. Salaun, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra, pp. 43–51 (June 2005)
23. Saltzer, J., Schroeder, M.: The Protection of Information in Computer Systems. Proceedings of the IEEE 63(9), 1278–1308 (1975)
24. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (abstract). In: PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, p. 188. ACM, New York (1998)
25. Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)

Revocation Schemes for Delegation Licences

Meriam Ben-Ghorbel-Talbi[1,2], Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], and Adel Bouhoula[2]

[1] Institut TELECOM/TELECOM Bretagne, 2 rue de la Chtaigneraie, 35576 Cesson Sévigné Cedex, France
{meriam.benghorbel,frederic.cuppens,nora.cuppens}@telecom-bretagne.eu
[2] SUP'COM Tunis, Route de Raoued Km 3.5, 2083 Ariana, Tunisie
bouhoula@planet.tn

Abstract. The paper presents revocation schemes in role-based access control models. We are particularly interested in two key issues: how to perform revocation and how to manage the revocation policy. We show how to deal with these two aspects in the delegation model based on the OrBAC formalism and its administration licence concept. This model provides means to manage several delegation types, such as the delegation or transfer of permissions and roles, multi-step delegation and temporary delegation. We state formally in this paper how to manage the revocation of these delegation schemes. Our model supports a wide spectrum of revocation dimensions such as propagation, dominance, dependency, automatic/user revocation, transfer revocation and role/permission revocation.

1 Introduction

In the field of access control, delegation is an important aspect and is managed as a special case of an administration task. An administration policy defines who is permitted to manage the security policy, i.e. who is permitted to create new security rules, or update or revoke existing security rules. A delegation policy defines who is permitted to manage existing security policies, i.e. who is permitted to delegate or revoke existing roles and privileges (e.g. permissions or obligations).

Many delegation schemes are defined in the literature, such as permanence, totality, monotonicity, multiple delegation and multi-step delegation. A complete access control model must provide a flexible administration model to manage these delegation aspects securely. But, it is also important to manage the revocation of such delegations. In [1] the authors propose a delegation model based on the OrBAC formalism and its administration licence concept [2]. This model provides means to deal with several delegation schemes thanks to the use of contextual and multi-granular licences. It supports the delegation and transfer of licences and roles, user-to-user and user-to-role delegation, permanent and temporary delegation, multiple delegation, simple and multi-step delegation, self-acted and agent-acted delegation. Moreover, several delegation constraints are

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 190–205, 2008.

specified thanks to the taxonomy of contexts defined in OrBAC, namely prerequisite, provisional, temporal, spatial.

We base our work on this delegation model and we focus on the revocation mechanism. Our approach provides means to express various revocation dimensions suggested in the literature [3,4] such as propagation, dominance, grant dependency, automatic revocation. We focus our work on two aspects of revocation. The first aspect is how to perform the revocation, i.e. automatically or manually, and the effect of the revocation on other delegations (e.g. cascade revocation). The second issue is how to manage the right of revocation, i.e. who is permitted to revoke delegations.

We show that using the contextual licences our model is more flexible and expressive. We deal with the different revocation schemes in a simple manner, and unlike the other work on revocation management, we do not need to define many kinds of revocation actions. Moreover, we can specify several revocation constraints that were not supported previously. Existing models only support constraints on the role membership of the grantor or the grantee, whereas, in our model we may specify several conditions using prerequisite, temporal, spatial, user-declared and provisional contexts and use them to specify contextual licences.

This paper is organized as follows. In section 2 we start with the system description. We give the basic concept of the delegation model and we introduce some definitions. In section 3 we present the revocation mechanism. This section focuses on how revocation is performed and managed in our model. Then, section 4 presents related work and shows the advantages of our approach. Finally, concluding remarks and future work are made in section 5.

2 System Description

In the model presented in [1] the administration and delegation privileges are managed in a homogeneous unified framework using the OrBAC formalism [5]. This model provides means to specify the security policy at the organization level that is independent of the implementation of this policy. Thus, instead of modelling the policy by using the concrete concepts of subject, action and object, the OrBAC model suggests reasoning with the roles that subjects, actions or objects play in the organization. The role of a subject is simply called a role, whereas the role of an action is called an activity and the role of an object is called a view.

In OrBAC, there are eight basic sets of entities: *Org* (a set of organizations), S (a set of subjects), A (a set of actions), O (a set of objects), R (a set of roles), $\mathcal{A}$ (a set of activities), V (a set of views) and C (a set of contexts). In the following we give the basic OrBAC built-in predicates:

- ***Empower*** is a predicate over domains *Org* x S x R. If *org* is an organization, s a subject and r a role, then *Empower(org, s, r)* means that *org* empowers subject s in role r.

- ***Use*** is a predicate over domains Org x O x V. If org is an organization, o is an object and v is a view, then *Use(org, o, v)* means that org uses object o in view v.
- ***Consider*** is a predicate over domains Org x A x $\mathcal{A}$. If org is an organization, α is an action and a is an activity, then *Consider(org, α, a)* means that org considers that action α implements activity a.
- ***Hold*** is a predicate over domains Org x S x A x O x C. If org is an organization, s a subject, α an action, o an object and c a context, *Hold(org, s, α, o, c)* means that within organization org, context c holds between subject s, action α and object o.
- ***Permission***, ***Prohibition*** and ***Obligation*** are predicates over domains Org x R_s x A_a x V_o x C, where R_s= $R \cup S$, $A_a = \mathcal{A} \cup A$ and V_o= $V \cup O$. More precisely, if org is an organization, g is a role or a subject, t is a view or an object and p is an activity or an action, then *Permission(org, g, p, t, c)* (resp. *Prohibition(org, g, p, t, c)* or *Obligation(org, g, p, t, c)*) means that in organization org, grantee g is granted permission (resp. prohibition or obligation) to perform privilege p on target t in context c.

The OrBAC model is associated with a self administrated model called AdOrBAC [2,6]. This model is based on an object-oriented approach, thus we do not manipulate privileges directly (i.e. *Permission*, *Prohibition* and *Obligation*), but we use objects having a specific semantics and belonging to specific views, called administrative views. Each object has an identifier that uniquely identifies the object and a set of attributes to describe the object. A view is used to structure the policy specification.

The management of delegation is based on the AdOrBAC model. Delegation is modelled just like the administration mechanism, using specific views called delegation views [1]. For the sake of simplicity, we assume that the policy applies to a single organization and thus we omit the Org entity in the following.

Fig.1 is an outline of OrBAC views. The view ***Licence*** is used to specify and manage users privileges: permissions, prohibitions and obligations. In the following we only consider licences interpreted as permissions. Objects belonging to this view have the following attributes: *grantee*: subject to which the licence is granted, *privilege*: action permitted by the licence, *target*: object to which the licence grants access and *context*: specific conditions that must be satisfied to use the licence. The existence of a valid licence is interpreted as a permission by the following rule:

$$\begin{aligned}&Permission(Sub, Act, Obj, Context)\text{:-}\\&\quad Use(L, licence), Grantee(L, Sub), Privilege(L, Act),\\&\quad Target(L, Obj), Context(L, Context).\end{aligned}$$

The view ***Role_assignment*** is used to manage the assignment of subject to roles. Objects belonging to this view are associated with the following attributes: *assignee*: subject to which the role is assigned and *assignment*: role assigned by the role assignment. There is the following rule to interpret the objects of the *role_assignment* view:

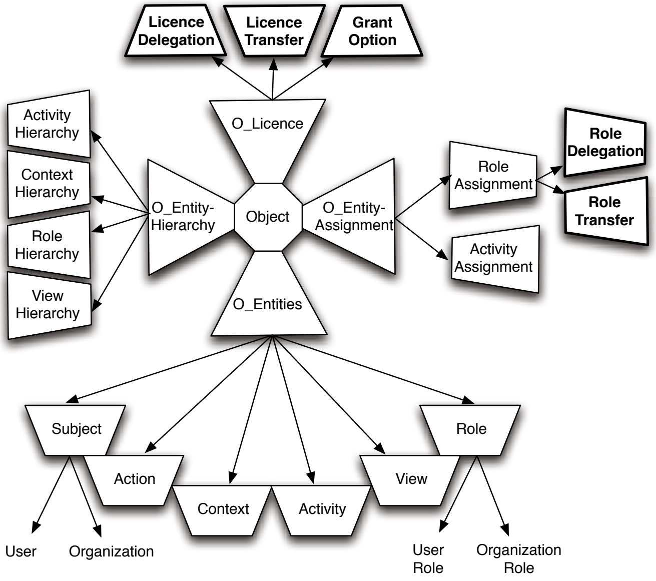


Fig. 1. OrBAC views

$$\begin{array}{l} Empower(Subject, Role)\text{:-} \\ \quad Use(RA, role\_assignment), Assignee(RA, Subject), Assignment(RA, Role). \end{array}$$

The views boldfaced in Fig. 1 are delegation views. ***Role_delegation*** and ***Role_transfer*** are two sub-views of *Role_assignment* defined to manage role delegation and transfer. ***Licence_delegation***, ***Licence_transfer*** and ***Grant_ option*** are three sub-views of *Licence* defined to manage licence delegation and transfer. Licence delegation concerns only *Permission* and *Obligation*. Thus, the delegation of negative privileges in not supported, although the administration model supports the granting of prohibitions. In fact, defining prohibitions is considered an administration task and it is not meaningful to delegate a prohibition.

Due to space limitation, we only consider in this paper *Licence_delegation* and *Grant_ option* views. Objects belonging to these have the same attributes and semantics as objects belonging to the *Licence* view (i.e. an assignment of a permission to a role or to a user). These objects also have an additional attribute called *grantor*: subject who delegates the licence. This attribute is important to revoke the delegation and to recover the grantor privileges when the transfer is revoked.

Note that there is a distinction between the delegation of a right R and the delegation of the right to delegate R, and for that purpose two delegation views are used. Objects belonging to the view *Licence_delegation* are interpreted as a delegation of simple rights, and objects belonging to the *Grant_option* view are interpreted as a delegation of delegation rights. This means that the privilege and the target of these objects concern delegation activities and delegation views, respectively.

Objects belonging to *Grant_option* have another additional attribute called *step*: the depth of the delegation chain. This attribute is used to control the propagation of the right to delegate. When the step is greater than 1 the grantee is allowed to propagate the delegation right (i.e. the right to delegate a licence or a role) to another user. Thus, a new permission is derived to the grantee as follows:

$$\begin{aligned}&Permission(GR, delegate, grant\_option, C \ \& \ valid\_redelegation(LG))\text{:-}\\&\quad Use(LG, grant\_option), Grantee(LG, GR),\\&\quad Context(LG, C), Step(LG, N), N > 1.\end{aligned}$$

Note that, the context *valid_redelegation* is used to control the scope of the redelegation. Thus the grantee is allowed to delegate only the right (or a sub-right of the right) he was received and with a lower step. So, we can be sure that the redelegated right will never be higher than the original right delegated by the first grantor. This context is defined as follows:

$$\begin{aligned}&Hold(U, delegate, LG', valid\_redelegation(LG))\text{:-}\\&\quad Use(LG', grant\_option), Sub\_Licence(LG', LG),\\&\quad Step(LG', N'), Step(LG, N), N' < N.\end{aligned}$$

where the predicate *Sub_Licence* is defined as follows:

$$\begin{aligned}&Sub\_Licence(L, L')\text{:-}\\&\quad Target(L, T), Target(L', T'), Sub\_Target(T, T'),\\&\quad Privilege(L, P), Privilege(L', P'), Sub\_Privilege(P, P'),\\&\quad Context(L, C), Context(L', C'), Sub\_Context(C, C').\end{aligned}$$

O is considered a Sub_Target of O' if they are two views and O is a Sub_View of O', or if O is an object used in view O', or if they are equal (see [7] for more details about the OrBAC hierarchy):

$$\begin{aligned}&Sub\_Target(O, O')\text{:-}\\&\quad Sub\_View(O, O'); Use(O, O'); O = O'.\end{aligned}$$

Predicates *Sub_Privilege* and *Sub_Context* are defined in a similar way.

2.1 Notation and Definitions

As we mentioned in the previous section, there is a distinction between the delegation of simple rights and the delegation of delegation rights. For this purpose, we denote L a simple delegated licence, i.e. objects belonging to the view *Licence_delegation*, and we denote LG a delegation licence, i.e. objects belonging to the view *Grant_option*.

Let O_L be the set of simple licences, O_{LG} the set of delegation licences and $O_{L_d} = O_L \cup O_{LG}$ the set of all delegated licences *Ld*.

We give now some definitions to deal with revocation in our model.

Definition 1. *Derivation relation.*

We distinguish between two derivation relations according to whether the type of derived licence is a simple or a delegation licence.

1. $\forall Ld, Ld' \in O_{LG}$, *Ld is* ***derived*** *from* Ld', *if:*
   - *the grantor of the licence Ld is the grantee of the licence* Ld',
   - *the licence Ld is a sub-licence of* Ld',
   - *the delegation step of Ld is lower than the delegation step of* Ld'.
2. $\forall Ld \in O_L, Ld' \in O_{LG}$, *Ld is* ***derived*** *from* Ld', *if:*
   - *the grantor of the licence Ld is the grantee of the licence* Ld',
   - *the licence Ld corresponds to the target definition of* Ld' *(i.e. Use(Ld, Target(Ld'))).*

Definition 2. *Delegation chain.*

1. *For each delegation licence* $LG \in O_{LG}$ *we can generate a delegation chain, which we call* $DC(LG)$*. A delegation chain is represented by a directed graph (see Fig 2.a). The nodes contain licences* $Ld \in O_{Ld}$*, and we denote* $N(Ld)$ *the node containing licence Ld. There is an arc from node* $N(Ld_1)$ *to node* $N(Ld_2)$*, if* Ld_2 *is derived from* Ld_1*. A node containing a simple licence* $L \in O_L$ *is always a leaf of the graph.*
2. *A node is rooted if it contains a licence that cannot be derived from any other licences. A delegation chain of a licence Ld is rooted if Ld is rooted.*
3. *When a node* N_i *is deleted (i.e. the licence contained in this node is revoked), a special arc labelled with a * is used to connect nodes* N_{i-1} *to* N_{i+1} *(see Fig2.b). In addition, we denote DC*(Ld) the delegation chain DC(Ld) that includes labelled arcs.*

Note that the delegation chain DC* is used to ensure that every delegated licence has a path that links it to the licences from which it is indirectly derived, even if some licences belonging to the delegation chain are removed. For instance, if we consider the delegation chain given in Fig. 2.b, then $DC(LG_1)=\{LG_2, L_3\}$ and $DC^*(LG_1)=\{LG_2, L_3, L_5\}$.

Definition 3. *Dependency.*

A licence Ld depends exclusively on a user U if there is no rooted delegation chain DC such that $Ld \in DC$ *and* $\forall Ld' \in DC$, $Grantor(Ld') \neq U$.

We assume that Dependent(Ld, U) is a predicate meaning that licence Ld depends exclusively on user U.

Theorem 1. *The delegation chains are computable in polynomial time.*

Proof. The delegation chain is based on the OrBAC model and its self-administration model. Policies associated with both of them can be expressed as recursive rules corresponding to a stratified Datalog program; the delegation chains are then obtained by computing a fixed point which is tractable in polynomial time. □

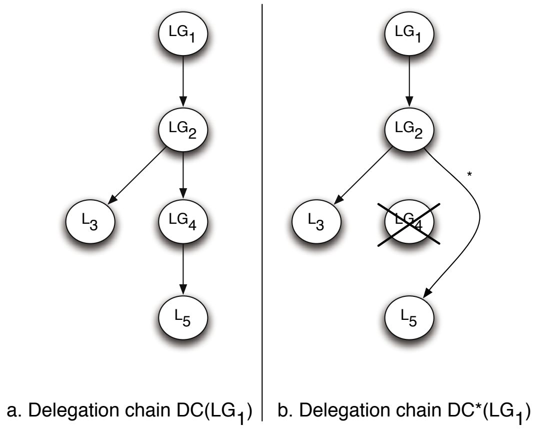


Fig. 2. Delegation Chain

3 Revocation

We focus in this section on two aspects of delegated right revocation. The first aspect is how to perform revocation. In our model revocation can be performed automatically when the delegation context does not hold, or manually by an authorized user. Following the classification defined in [4], user revocation can be categorized into three main dimensions: propagation, dominance and resilience. Propagation concerns the management of the revocation in the case of multi-step delegation, dominance concerns the revocation impact on other delegations associated with the same grantee, and resilience concerns the persistence of the revocation in time, i.e. the revocation by removal of the permission (non-resilient) or by adding a negative permission that has higher priority (resilient). In our delegation model we do not consider resilient revocation. Indeed, we consider that delegation concerns only positive permissions and granting prohibitions is an administrative task only.

Obviously, to perform a revocation the user must have the permission to revoke the delegation. This is the second issue of revocation: how to manage the right of revocation. For this purpose, we consider the grant dependency characteristic. Grant Dependent (GD) revocation means that the user can only revoke his/her delegations. Grant Independent (GI) means that the delegation can be revoked by any authorized user. For instance, a user empowered in a role R can revoke any grantee empowered to this role by delegation.

3.1 User Revocation

As mentioned in the previous section, delegation is performed by adding objects into delegation views. The existence of these objects is then interpreted as a permission. The revocation of the delegation is performed by removing these objects from the delegation views. Each object is associated with a grantor attribute that indicates the subject that is performing the delegation. By default, each object can be removed by his/her grantor. This can be specified implicitly or explicitly in our model (see section 3.3). Thus, we can ensure that each delegation can be revoked.

The effect of the licence revocation on the other delegations depends on the revoker needs. He/she can choose, for instance, to revoke the whole delegation chain (cascade revocation) or to revoke all other delegations associated with the same grantee (strong revocation).

Propagation. In the case of multi-step delegation, the delegation of each delegation licence (i.e. using the view *Grant_option*) can generate a delegation chain. Hence, the revoker can choose to revoke only the licence he/she has delegated or also all licences derived from it.

Definition 4. *Simple revocation.*

This is the simplest revocation scheme. The revocation involves deleting the licence from the delegation view and does not affect the other delegated licences. We formally define the request to revoke a delegation as follows:

```
Request(U, revoke, L) then
   forall $L_i \in$ Derived_licences(L) and $L_j \in$ Parent_licences(L) do
     Add a labeled arc from $N(L_j)$ to $N(L_i)$,
     Remove(L).
end
```

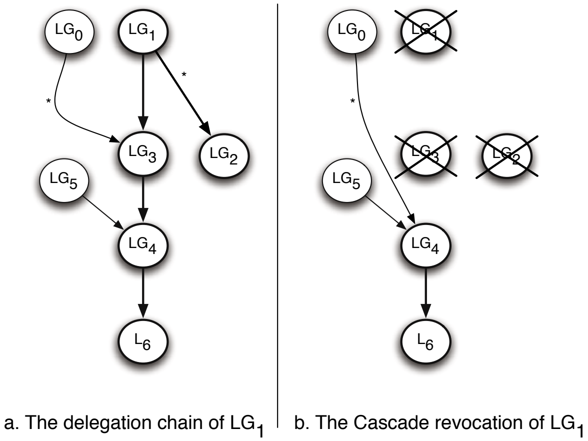
We assume that, Derived_licences(L) is the set of all licences L' such that there is an arc (labelled or not labelled) from node $N(L)$ to node $N(L')$ and Parent_licences(L) is the set of licences L'' such that there is an arc (labelled or not labelled) from node $N(L'')$ to node $N(L)$.

Definition 5. *Cascade revocation.*

This involves the revocation of all the licences belonging to the delegation chain. But, this revocation should not affect the licences belonging to the delegation chain (DC) of other licences. The reason is that if the grantor of a licence L has received the permission to delegate this licence from two or more different delegation licences, then if one of these delegations is revoked, the grantor still has the right to delegate L (an illustrative example is given below). In our model, cascade revocation is defined as follows:

```
Request(U, Cascade_revoke, L) then
  Request(U, revoke, L),
   forall $L_i \in$ Derived_licences(L) do
      if $L_i \notin DC(L'), L' \in O_{LG}$ then
         Request(U, Cascade_revoke, $L_i$).
     end
end
```

Example 1. We consider the example shown in Fig.3, and we assume that LG_1 is revoked with the cascade option. On the first pass, licence LG_1 is removed. On the second pass, LG_2 and LG_3 are revoked and a labelled arc is added to connect node LG_0 to LG_4. Note that LG_3 is revoked because it belongs to DC*(LG_0) and not to $DC(LG_0)$. Finally, the cascade revocation process is stopped because LG_4 belongs to $DC(LG_5)$.



a. The delegation chain of LG_1 b. The Cascade revocation of LG_1

Fig. 3. Cascade revocation

Dominance. The revocation described so far is a weak revocation. This means that the revocation of a licence L will only affect this licence (in the case of simple revocation) or the licences belonging to its delegation chain (in the case of cascade revocation). However, a strong revocation will affect the other licences associated with the same grantee.

Definition 6. *Strong revocation.*
The strong revocation of licence L means that all the licences equal to L (or are a sub-licences of L) that depend on the revoker and associated with the same grantee must be revoked. This is defined as follows:

```
Request(U, Strong_revoke, L) then
Request(U, revoke, L),
forall $Ld_i \in O_{Ld}$ do
    if Grantee($Ld_i$) = Grantee(L) and Sub_licence($Ld_i$,L) and
    Dependent($Ld_i$,U) then
        Request(U, revoke, $Ld_i$).
    end
end
```

As mentioned in Definition 3, $Dependent(Ld, U)$ is a predicate meaning that licence Ld depends exclusively on user U.

Definition 7. *Strong-Cascade revocation.*

We consider the strong-cascade revocation of a licence L as a strong revocation of all licences belonging to the delegation chain of L. This is defined as follows:

```
Request(U, Strong_Cascade_revoke, L) then
Request(U, Strong_revoke, L)
forall $L_i \in$ derived_licences(L) do
    if $L_i \notin$ DC(L'), $L' \in O_{LG}$ then
        Request(U, Strong_Cascade_revoke, $L_i$)
    end
end
```

Example 2. We consider the example shown in Fig.4, where we represent in each node the delegated licence, the grantor and the grantee of this licence. We assume that licence LG_5 is a sub-licence of LG_3 and LG_7 is a sub-licence of LG_4.

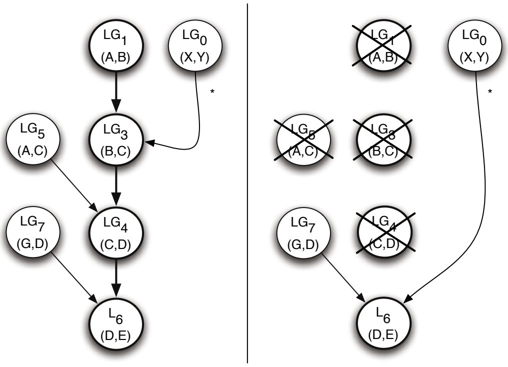


Fig. 4. Strong-Cascade Revocation

If A the grantor of licence LG_1 revokes this licence with the strong-cascade option, then the whole delegation chain of LG_1 will be revoked with the strong cascade option as well. On the first pass, licence LG_1 is revoked by A. On the second pass licence LG_3 is revoked and also licence LG_5 since it is a sub_licence of LG_3 and it depends on A. On the third pass, LG_4 is revoked but not LG_7 because it is independent of A. L_6 is not revoked because it belongs to $DC(LG_7)$.

Theorem 2. *The revocation requests are computable in polynomial time.*

Proof. A request to revoke a licence requires a recursive search in the delegation chain of this licence, therefore using theorem 1, it is computable in polynomial time. □

Other revocation schemes. The revocation discussed in this section only concerns delegated licences, however the delegation model supports both role and licence delegation. Role revocation is omitted for the sake of simplicity, but we can deal with this aspect similarly as described above. We have simply to consider the set of delegated roles, O_R (the set of objects belonging to the view *Role_delegation*) and we replace the set O_L (the set of delegated licences) by $O_D = O_L \cup O_R$ (the set of delegations). The remainder is unchanged, except for strong revocation where *sub_licence* is replaced by *sub_role* in the case of role revocation.

Additionally, the delegation model supports role and licence transfer. But there is no need to use specific functions to deal with the transfer revocation. In fact, the grantor automatically recovers his/her rights when the transfer is revoked (see [1] for more details). As we have mentioned earlier, when the licence is removed the permission delegated to the grantee is no longer derived, but in the case of transfer, the prohibition associated with the grantor is no longer derived either.

3.2 Automatic Licence Revocation

In our model, the licence revocation can also be performed automatically using the notion of context. In fact and contrary to other existing models (see section 4), *Licences* are associated with an attribute called *Context* used to specify conditions (see [8] for more details). Concrete permissions are derived from *Licences* only if the context is active. Hence, the delegation is revoked automatically if the context does not hold. Note that in this case the licence is not removed as in user revocation, but the permission is not derived.

For instance, we consider the case when the user U_1 delegates licence L to user U_2 with the following attributes: Grantor: u_1, Grantee: u_2, Privilege: *read*, Target: $file_1$, Context: *weekend*. This means that the following permission is derived for user U_2: $Permission(u_2, read, file_1, weekend)$.

This permission is derived only if the context *weekend* holds, thus the user U_2 is allowed to read the file $file_1$ only during the weekend. We can say that the delegation is revoked automatically during the other days of the week.

Note that in the case of multi-step delegation, when a delegation licence LG is revoked automatically (i.e. the delegation context of LG does not hold), then all the delegation licences derived from it are revoked automatically as well (i.e. the delegation context of these licences does not hold as well). This is due to the fact that when we derive a licence LG' from LG, then LG' must be a sub-licence of LG. Therefore, the delegation context of LG' must be equal to or a sub-context of the delegation context of LG.

Moreover, in the case of licence transfer, the grantor revocation is done automatically using contextual prohibition. In fact, when a transfer is performed, a prohibition associated with the highest priority level is automatically assigned to the grantor while the delegation is active (see [1]). Therefore, the grantor will lose the permission he/she has delegated. The context of the prohibition and of the delegated permission are the same. So, when the delegation context does not hold, the prohibition is no longer derived and the grantor will automatically retrieve his/her permission.

3.3 Managing Revocation

A revocation policy defines who is permitted to revoke delegations (roles or licences). In the literature some models [9,10,11,12] consider that the right to revoke his/her delegations is implicitly defined, i.e. the grantor of the right R is automatically allowed to revoke it. Other models [13,14,15] use specific functions to manage the revocation policy, such as *can_revoke*, *can_revokeDG*, *can_revokeGI*, *can_u2u_revokeGD*. This approach is more general since it dissociates the right to revoke from the right to delegate a privilege. Therefore, it supports more revocation schemes such as Grant Independent.

In our model, we follow the second approach, but we do not use specific functions to define the right to revoke. In fact, we manage delegation and revocation by the assignment of subjects (users or roles) to permissions just like in the AdOrBAC model. These permissions apply to the activities *delegate* and *revoke*, and are associated with a delegation target (the delegation views). So, they allow users to add and remove objects from the delegation views.

Note that using the derivation rules, we can implicitly specify that grantors who are permitted to delegate a right are also permitted to revoke this right. But in addition, we can specify explicitly who is permitted to revoke delegations and, thanks to the use of contexts, we can specify different conditions to restrict the revocation rights. In the following we give some context examples dealing with Grant Dependent and Independent revocation.

Example 3. Grant Dependent revocation.

In the case of Grant Dependent revocation only the grantor is allowed to revoke the delegated licence. To deal with this aspect we define the prerequisite contexts gd_L as follows[1]:

$$Hold(U, revoke, L, gd_L)\text{:-}$$
$$Use(L, licence\_delegation); Use(L, grant\_option), Grantor(L, U).$$

Using this context, the administrator can specify that all users are authorized to revoke their delegated licences. This is defined by the following:

$$Permission(default\_Role, revoke, licence\_delegation, gd_L).$$

$$Permission(default\_Role, revoke, grant\_option, gd_L).$$

where *default_Role* is a role to which all authorized users are empowered.

We can also consider the case of role revocation as follows:

$$Hold(U, revoke, RD, gd_R)\text{:-}$$
$$Use(RD, role\_delegation), Grantor(RD, U).$$

Similarly, the administrator can specify that all users are authorized to revoke their delegated roles as follows:

$$Permission(default\_role, revoke, role\_delegation, gd_R).$$

Example 4. Grant Independent revocation.

[1] The operator ';' corresponds to a disjunction.

We can specify many contexts to deal with this aspect according to the administrator's needs. Contexts provide our model with high flexibility and expressiveness.

For instance, we may consider that a user can revoke a licence (respectively a role) if this licence (or this role) is derived from this user but depends exclusively on him/her. Hence, he/she cannot revoke a licence that also depends on other users. We define for this purpose the contexts *Ancestor Dependent* ad_L and ad_R to revoke a licence and a role, respectively:

$Hold(U, revoke, L, ad_L)$:-
 $Use(L, licence\_delegation); Use(L, grant\_option), Dependent(L, U)$.

$Hold(U, revoke, RD, ad_R)$:-
 $Use(RD, role\_delegation), Dependent(RD, U)$.

We can also define the context *Role Dependent (rd)* to say that any user empowered in role R can revoke this role:

$Hold(U, revoke, RD, rd)$:-
 $Use(RD, role\_delegation), Assignment(RD, R), Empower(U, R)$.

Note that to manage the revocation policy, the administrator only specifies who is permitted to perform a simple revocation. We assume that the right to perform the other revocation schemes, namely cascade, strong and strong-cascade revocation, is automatically derived from the simple revocation. For instance, the *request (U, cascade-revoke, L)* is considered as a set of simple requests to revoke the licences that belong to the delegation chain of L. Hence, if the user is allowed to revoke these licences then he is allowed to revoke L with the cascade option. A straightforward refinement of our model involves specifying that the revocation request is atomic or not, i.e. if the user is not allowed to revoke all the related licences, then we have to specify if the request to revoke with the cascade, strong or strong-cascade option, can be partially accepted or must be totally rejected.

Moreover, using the contextual licences, we can specify complex security rules to manage revocation. Indeed, we can define several conditions using the taxonomy of contexts (e.g. temporal, prerequisite, provisional [8]). These conditions may concern the grantor or the grantee attributes, previous actions, the delegated right (i.e. the role, the target, the privilege), the time, circumstances (e.g. urgency). For instance, we can specify that a given user U_1 is allowed to revoke the role R transferred by the grantor U_3 to the grantee U_2, only if U_3 is not on vacation. We can also define that U_1 can revoke U_2 if this user has performed a given action A in a given object O. We may also specify that U_1 is authorized to revoke U_2, if U_2 is the assistant of U_1, or is associated with the same department as U_1. These kind of conditions are not supported by the existing models since they only specify constraints on the role membership of the grantor or the grantee.

4 Related Work

In [4] authors classify the revocation into three dimensions: resilience, propagation and dominance. These dimensions are combined to provide eight different revocation schemes. This paper proposes to study permission revocation in a generic access control framework with a grant for both positive and negative permissions, where negative permissions dominate positive ones. The concept of inactive permissions is used to deal with resilient revocation, namely, positive permissions are inactivate when a negative permission is granted.

[9] addresses the revocation of certificates. The proposed framework supports revocation schemes such as propagation, dominance and grant dependency. Strong revocation means that the user is permitted to revoke any certificate that belongs to the delegation chain of a certificate issued by him/her. This corresponds to ancestor-dependent revocation in our paper.

Work [3,10,11,12,13,14,15] deals with revocation in role-based access control models. RBDM0 [3] is the first delegation model of Barka and Sandhu. It addresses role revocation and supports automatic revocation using a time out, and grant independent revocation (which corresponds to role-dependent revocation in our paper). RBDM1 [13] is an extension of this model, which supports cascade revocation, strong revocation, grant-dependent and role-dependent revocation. The strong revocation of a role is considered as a revocation of both explicit and implicit memberships (i.e. all roles junior to that role are revoked), this is a specific case of strong revocation presented in our paper. Role-role revocation is defined using the relation: *Can-Revoke* $\subseteq R$ x R. *Can-Revoke(x, y)* $\in$ *Can-Revoke* means that the user empowered in role x can revoke the membership of the delegate member y in role x. RDM2000 [15] is based on RBDM0. It supports cascade and strong revocation (strong option has the same definition as in RBDM0). Revocation is managed using the following relations: *Can-RevokeGD* $\subseteq R$ to manage the grant-dependent revocation and *Can-RevokeGI* $\subseteq R$ to manage the role-dependent one.

The two models [11,12] address the revocation of permissions in workflow systems. The first model supports automatic revocation using the notion of case (an instance of a workflow process) and grant-dependent revocation (the right to revoke his/her own delegations is implicitly defined). It also supports cascade revocation and uses a delegation graph similar to our model, namely nodes represent accepted delegations. The second model deals with automatic revocation using lifetime, cascade revocation and dependency. Users are automatically allowed to revoke their own delegations, and the grant-independent option is considered as revocation by the security administrator.

Compared to these works, our model is more expressive since it supports various revocation schemes such as automatic/manual, simple/cascade, weak /strong, grant dependent/independent (e.g. role dependent, ancestor dependent). Only resilient revocation is not considered because we assume that granting negative permissions is an administration task only. These different schemes apply to the revocation of delegation/transfer of roles/permissions. Moreover, thanks to the use of contexts, our model is more flexible and simpler to manage.

Namely, there are no specific permissions or actions for each revocation task like in related work. Hence, we have simply to define contexts to specify the different revocation schemes. On the other hand, using contextual permissions, we can specify several constraints to manage revocation that are not supported by other work. Indeed, they only support constraints on the role membership of the grantor or the grantee, whereas in our model we can specify several kinds of conditions concerning grantor/grantee attributes, previous actions, delegated rights, the time.

5 Conclusion

In this paper we have proposed to deal with revocation in role-based access control models. Our work is based on the delegation model presented in [1]. This model is flexible and various delegation schemes are defined.

Our revocation study has focused on two issues: how to perform revocation and how to manage the revocation policy. Revocation can be performed automatically when the delegation context does not hold, or manually by an authorized user. The effect of revocation on other delegations varies according to the revoker's needs. The revocation can be with a cascade so all the delegation chains are revoked, or with a strong option so all the delegations assigned to the grantee are revoked. We can also combine these two features to revoke with a strong-cascade option.

To manage the revocation policy, we have to specify who is permitted to delete objects from the delegation views. Thanks to the use of contexts we can specify various conditions to restrict the revocation rights. We have given some context examples to specify that all grantors are permitted to revoke their delegations, or all users empowered in a role R are permitted to revoke R, or finally, all users are permitted to revoke licences (or roles) that belong to their delegation chain.

Although many revocation schemes are supported, such as propagation, dominance, automatic revocation, grant dependency, transfer revocation, we have dealt with these different revocations in a simple manner and there is no need to deal with each level separately. Thanks to the facilities provided by the OrBAC model (i.e. contextual licences, the use of views), we do not use specific functions to manage revocation like in other related work. Therefore, it is easier to extend our study to deal with other revocation schemes and constraints, since we have simply to define new contexts.

In this paper we have focused on the delegation and revocation of licences that are interpreted as permissions. Future work will be to enrich our model to study the delegation and revocation of obligations.

References

1. Ben-Ghorbel-Talbi, M., Cuppens, F., Cuppens-Boulahia, N., Bouhoula, A.: Managing Delegation in Access Control Models. In: Proceedings of the 15th International Conference on Advanced Computing and Communications (ADCOM 2007), Guwahati, Inde, pp. 744–751. IEEE Computer Society Press, Los Alamitos (2007)

2. Cuppens, F., Cuppens-Boulahia, N., Coma, C.: Multi-Granular Licences to Decentralize Security Administration. In: Proceedings of the First international workshop on reliability, availability and security (SSS/WRAS 2007), Paris, France (November 2007)
3. Barka, E., Sandhu, R.: A Role-based Delegation Model and Some Extensions. In: Proceedings of the 23rd National Information Systems Security Conference (NISSC 2000), Baltimore, MD (October 2000)
4. Hagström, Å, Jajodia, S., Parisi-Persicce, F., Wijesekera, D.: Revocation - a Classification. In: Proceedings of the 14th Computer Security Foundation Workshop (CSFW 2001), Cape Breton, Nova Scotia, Canada, IEEE Computer Society, Los Alamitos (2001)
5. Abou-El-Kalam, A., Benferhat, S., Miège, A., Baida, R.E., Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y., Trouessin, G.: Organization Based Access Control. In: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003). IEEE Computer Society, Los Alamitos (2003)
6. Cuppens, F., Miège, A.: Administration Model for Or-BAC. International Journal of Computer Systems Science and Engineering (CSSE) 19(3) (May 2004)
7. Cuppens, F., Cuppens-Boulahia, N., Miège, A.: Inheritance Hierarchies in the Or-BAC Model and Application in a Network Environment. In: Proceedings of the 3rd Workshop on Foundations of Computer Security (FCS 2004), Turku, Finland (July 2004)
8. Cuppens, F., Cuppens-Boulahia, N.: Modeling Contextual Security Policies. International Journal of Information Security (November 2007)
9. Firozabadi, B.S., Sergot, M.: Revocation Schemes for Delegated Authorities. In: Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (2002)
10. Nguyen, T.A., Su, L., Inman, G., Chadwick, D.: Flexible and Manageable Delegation of Authority in RBAC. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW 2007). IEEE Computer Society, Los Alamitos (2007)
11. Wainer, J., Kumar, A., Barthelmess, P.: DW-RBAC: A Formal Security Model of Delegation and Revocation in Workflow Systems. Information Systems 32(3), 365–384 (2007)
12. Wei, Y., Shu, Q.: A Delegation-Based Workflow Access Control Model. In: Proceedings of the First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007). IEEE Computer Society, Los Alamitos (2007)
13. Barka, E., Sandhu, R.: Role-Based Delegation Model/ Hierarchical Roles (RBDM1). In: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004), Tucson, Arizona (December 2004)
14. Lee, Y., Park, J., Lee, H., Noh, B.: A Rule-Based Delegation Model for Restricted Permission Inheritance RBAC. In: Proceedings of the 2nd International Conference (ACNS 2004), Yellow Mountain (June 2004)
15. Zhang, L., Ahn, G.-J., Chu, B.-T.: A Rule-Based Framework for Role-Based Delegation and Revocation. ACM Transactions on Information and System Security (TISSEC) 6, 404–441 (2003)

Reusability of Functionality-Based Application Confinement Policy Abstractions

Z. Cliffe Schreuders and Christian Payne

School of IT, Murdoch University, South Street Murdoch WA 6150, Australia
{c.schreuders,c.payne}@murdoch.edu.au

Abstract. Traditional access control models and mechanisms struggle to contain the threats posed by malware and software vulnerabilities as these cannot differentiate between processes acting on behalf of users and those posing threats to users' security as every process executes with the full set of the user's privileges. Existing application confinement schemes attempt to address this by limiting the actions of particular processes. However, the management of these mechanisms requires security-specific expertise which users and administrators often do not possess. Further, these models do not scale well to confine the large number of applications found on functionality-rich contemporary systems. This paper describes how the principles of role-based access control (RBAC) can be applied to the problem of restricting an application's behaviour. This approach provides a more flexible, scalable and easier to manage confinement paradigm that requires far less in terms of user expertise than existing schemes. Known as functionality-based application confinement (FBAC), this model significantly mitigates the usability limitations of existing approaches. We present a case study of a Linux-based implementation of FBAC known as FBAC-LSM and demonstrate the flexibility and scalability of the FBAC model by analysing policies for the confinement of four different web browsers.

Keywords: Functionality-Based Application Confinement (FBAC), Role-Based Access Control (RBAC), Application-Oriented Access Control, Application Confinement, Sandbox, Usable Security, Reusable Policy.

1 Introduction

Existing widely used access control models reflect the traditional paradigm of protecting users from one another. Although user-oriented access control models such as traditional mandatory access control (MAC), discretionary access control (DAC) and role-based access control (RBAC) restrict the actions of users, these are generally not able to distinguish between an application performing legitimate actions on behalf of a user and code that is using these privileges nefariously. As a result, programs are essentially fully trusted: once executed malicious code typically has complete access to the user's privileges.

Application confinement models have been developed to restrict the privileges of processes, thereby limiting the ability of these programs to act maliciously. However, established application confinement models that allow finely-grained control

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 206–221, 2008.

over access to resources require the construction of extremely complex policies [1, 2]. These require significant technical expertise to develop and have limited scalability as confining each application involves the construction of a new detailed policy. Unfortunately, to date this has limited their practical usefulness and acceptance.

By recognizing that the goal of restricting users is essentially analogous to that of restricting applications, it follows that the principles from existing user-oriented access control models may be applied to the problem of process confinement. Specifically, applying the principles of role-based access control to application confinement leverages the flexibility and efficient management of RBAC to provide hierarchical policy abstractions for restricting applications, which eases policy development and association. These constructs can be parameterised to provide flexible and reusable application-oriented policy abstractions for improved usability, manageability, scalability and security.

2 Background

2.1 Application Confinement

A number of application confinement models have been developed to provide restricted environments for applications, thereby limiting their ability to behave maliciously. Some simply isolate programs to a limited namespace using a traditional sandbox [3] or virtual machine [4], examples include `chroot()`, FreeBSD jails [5] Solaris Zones [6], and Danali [7]. Others allow restricted access to selected shared resources, such as the Java [8] and .NET [9] sandboxes where applications are restricted by complex administrator-specified policies based on the properties of the code. Some models exist based on the paradigm of label-based integrity preservation where subjects are labelled high or low in integrity and the flow of information between levels serves as the basis for policy [10, 11]. Other restricted environments require the specification of a detailed policy detailing each application's access to specific resources. This applies to confinement mechanisms including Janus [12], Systrace [13], Novel AppArmour [14] (previously known as SubDomain), TRON [15], POSIX capabilities [16], Bitfrost [17], CapDesk [18], and Polaris [19]. Methods of mediating this type of access control include using capabilities [20] or system call interposition [2]. Other schemes, such as domain and type enforcement (DTE) [21] and the Role Compatibility model [22] allow the definition of multiple restricted environments, and propagating processes transition between them.

Unfortunately all of these mechanisms have limitations and problems. Isolation-based approaches typically involve significant redundancy as shared resources must be duplicated and they also severely limit the ability of applications to exchange data with one another [23]. On the other hand, more finely-grained restricted access control policies are difficult and time-consuming to define and manage. The task of translating high level security goals into finely grained policies is problematic, making these policies difficult to both construct and verify for completeness and correctness [1, 24]. Furthermore, once constructed an individual policy will apply primarily to only a single application, meaning that the work involved in constructing suitable policies for all necessary applications is considerable. For example, specifying DTE domain policies is complex and although multiple processes can be confined by a single domain, domains must be specified separately [25]. There is significant overlap of privileges granted to compiled domain policies, and typically any non-trivial application is assigned a separate domain. Finally, specifying file and domain transitions can also be a complex task as programs need specific authorisation to label files as being accessible to programs in different domains, and users and programs both need permission in order to execute programs belonging in another domain [26].

These application confinement schemes lack flexible policy abstractions which can allow application access policies to be meaningfully reused while providing fine grained restrictions. With isolation sandboxes the container itself acts as the only policy abstraction – a simple collection of subjects and resources. Existing schemes which mediate finely-grained privileges to applications are generally either devoid of policy abstraction (a list of privileges are directly associated with a program) or contain large monolithic self-contained abstractions (such as DTE domains or RC roles) which cannot be flexibly reused for different applications unless they share the exact same privilege requirements. As a result, these application confinement architectures do not provide a practical or scaleable solution for conveniently confining multiple applications.

While a few implementations of these models allow policy abstractions to be comprised of smaller components they are reduced to a single monolithic policy abstraction before use, which limits their usefulness at run-time and their reusability. For example, SELinux's DTE Domain specification can include macros in the m4 language. Before policy is applied, these are expanded into many lines of rules granting all the required privileges. The result is a single domain with a fixed set of privileges, typically those required by a single program. Likewise, at system start-up abstractions in AppArmor application profiles are translated into a raw list of privileges associated with the program. This monolithic approach to policy abstraction also means that any finer grained abstractions which may have been used to construct policy are not available when managing the privileges of a process.

2.2 Role-Based Access Control (RBAC)

Role-based access control (RBAC) is a user-oriented access control model which associates users with privileges via organizational abstractions known as *roles* [27]. When a user joins an organisation they are assigned the roles representing the privileges required by their responsibilities and duties and this eliminates the task of manually assigning permissions to each new user [28]. Access decisions are then made based on the permissions associated with the roles the user is assigned. Policy reusability is enhanced through role hierarchies which allow roles to be defined in terms of other roles. Also, role constraints such as *separation of duty* can restrict certain conflicting permissions from being associated with the same user (*static separation of duty*) or accessed concurrently (*dynamic separation of duty*) [29].

Many similarities can be observed between the motivation for the development of RBAC in relation to traditional access controls and the current problems faced in the domain of application confinement. RBAC provides a conceptually-straightforward, scalable and abstract association between users and the privileges they require in order to perform their designated duties within an organisation. This highlights the advantages a model which provides similar abstract associations between applications and the privileges they require can provide to application-oriented access control.

3 Functionality-Based Application Confinement

3.1 Policy Abstraction

The notional similarities previously noted between user confinement via access controls and application confinement models suggest the applicability of traditional access control principles to the problem of restricting applications. In particular, many of the design principles of RBAC can be applied to manage the privileges of executing programs. Based on this, a model known as *functionality-based application confinement* (FBAC) has been developed [30]. Designed to be analogous to the specifications contained in the NIST/ANSI INCITS RBAC model [31, 32], FBAC acts as an additional layer above traditional access control models and treats all software that the user executes as untrusted by limiting its access to only the resources deemed necessary for the application to operate as required.

Application confinement policies can be defined in terms of their behavioural classes [33] which are conceptually analogous to RBAC roles. FBAC uses abstractions similar to RBAC roles and role hierarchies which are used to define complex, finely-grained application confinement policies in terms of high level abstractions. Consequently applications are confined to those resources deemed necessary by its assigned *functionalities*.

Functionalities are hierarchical policy abstractions which form the basis of FBAC policy. Functionalities can represent high-level behavioural classes of applications (for example, "Web_Browser" or "Web_Server") and these can inherit lower level functionalities that represent application functionality such as "http_client", "ftp_ client" and "read_files_in_directory". These functionalities are associated with privileges that are made up of *operations* on *objects*.

The RBAC model and an FBAC confinement are structurally analogous but very different in purpose. While RBAC is a user confinement model for system administrators to restrict what permissions users hold according to their duties within an organisation, FBAC is a framework for users to restrict the privileges of each application based on the functionality it provides.

Related to the concept of discretionary role-based access control (DRBAC) [34, 35] where users have the ability to define and activate their own RBAC roles, FBAC also applies RBAC concepts to allow users to confine themselves; however, FBAC is focused on restricting applications rather than users.

3.2 Parameterisation

While RBAC roles are self-contained with each user receiving the same set of privileges [36], in an application confinement context behavioural classes are better defined in terms of parameterised categories [33]. Unlike RBAC role associations, FBAC functionality associations are parameterised to allow functionalities to adjust to the needs of different applications. For example, although an application may be classified by a general grouping such as "Web_Server", in order to create an effective confinement policy certain application-specific details (such as the location of files and directories it uses) must still be defined.

FBAC provides parameterised functionalities to allow policies to be more precisely defined in terms of application-specific details. FBAC functionalities are passed arguments in a way similar to how subroutines are in programming languages. This allows the policy abstraction to be adapted to the specifics of individual applications providing related features. Functionality definitions can also contain default arguments which allow further ease of use in common cases without sacrificing flexibility. This means applications are defined in terms of functionalities plus any information required by those functionalities. Functionalities may use this information to inherit from other functionalities or define the resources associated with operations.

3.3 Mandatory and Discretionary Controls

Unlike existing application confinement schemes which are either applied as a discretionary control (such as Janus or TRON) or as a mandatory control (such as with DTE or AppArmor), FBAC supports both mandatory and discretionary access controls simultaneously. Administrators can specify policies which govern the behaviour of applications to enforce system-wide security goals, restrict users to particular programs, and manage user protection. Users may then further confine these applications to protect their own resources from malicious code.

This is achieved by layering FBAC confinements. A confinement may apply to multiple users and may reuse the functionalities from other confinements. The resulting authority granted to an application is the intersection of the confinements for that application which apply to the executing user. This layered approach to application confinement is unique and provides defence in depth while requiring the maintenance of only one mechanism. Because confinements can share the same functionalities this greatly reduces the overhead of managing multiple layers of application-oriented access controls, while enforcing the security goals of both users and administrators.

4 Defining and Managing Policy

The FBAC model greatly simplifies the management of application confinement policies compared with existing models. Functionalities are established representing the various functional requirements of applications. Privileges can be assigned to these functionalities directly and may also be inherited by other contained functionalities. The applications have these functionalities associated with them as required by their expected behaviour and when the program is executed, this will activate the functionalities that apply to it and thus define its privileges at runtime.

Initial policy definition in FBAC involves the creation of new functionalities in terms of low level privileges and existing functionalities, assigning the rights necessary for applications to function according to the behaviour described by functionalities. This is influenced by security goals and application behaviour and resource requirements. Although the design of FBAC significantly reduces the complexity of privilege assignment compared with other finely-grained confinement models, this initial process does require greater expertise than other aspects of the framework and may be completed by a trusted third party rather than by end users.

Once defined these functionalities may be reused by multiple users to restrict as many applications as appropriate. End users require little expertise to identify the functionalities relevant to their applications based upon the program's expected behaviour. These are then associated with the application and parameters are provided where necessary. This process is far simpler than with alternative confinement techniques where complex policies must be defined for each individual application.

5 Web Browser Case Study

A language for expressing FBAC policies has been developed and a prototype implementation of the model as a Linux Security Module (LSM) [37] called *FBAC-LSM* is near completion. The policy requirements for a number of applications were analysed and a hierarchal FBAC policy for FBAC-LSM has been created. We now present a case study of the application of FBAC policies to four common web browsers — Firefox, Opera, Epiphany and Lynx — for the purposes of demonstrating policy flexibility and reusability.

5.1 Restricting Applications

To confine a web browser such as Firefox using the graphical policy manager tool the user simply chooses the high level functionalities relating to that application's functionality. The user assigns a base functionality such as "Standard_ Graphical_ Application_Base" and any high level functionalities which describe what the application is expected to do (such as the "Web_Browser" functionality as defined in Figure 2 and provides application specific parameters such as where the program is installed, the location of its configuration files, where the program downloads files to, and potentially a list of hosts it can connect to. If confining a browser such as Opera that supports additional functionality, other corresponding high level functionalities are also assigned such as "Email_Client", "Irc_Chat_Client", "News_ Reader_ Client" and "BitTorrent_Client".

Unlike some operating systems where each application's files are typically found in a very small number of directories, Linux organises application files based upon the filesystem hierarchy standard (FHS). This can lead to an application's files being spread throughout the filesystem tree and in some cases parameter value specification may necessitate a degree of familiarity with this arrangement. Any complexity due to this can be mitigated by the use of parameter descriptions suggesting the location of files according to the FHS and the provision of a list of pathnames used by a program (for example, as in the case of Opera which provides this information to the user). Furthermore, techniques are currently being developed to automatically derive parameter values based on associated functionalities, and package management and filesystem analysis. A graphical policy management tool has been created which removes the need for end-users to be familiar with the FBAC-LSM policy language and policy association becomes a matter of pointing and clicking. However, even so, the FBAC-LSM policy language is simpler and provides greater abstraction than existing alternatives.

A FBAC policy for the Firefox browser created with the graphical policy manager tool is given in Figure 1. For comparison purposes, additional policies for the three other browsers considered in the case study are contained in Appendix A.

The Firefox policy from Figure 1 begins by specifying the executables which are used to run the application (binarypaths). Next it identifies the two functionalities that this application encompasses: “Standard_Graphical_Application_Base” and “Web_ Browser”. These functionalities are parameterised to address the specifics of the application, for example to specify where the various files it uses are located and the hosts to which it is permitted to connect. These parameters can easily be changed to grant the application access to different resources. For example, to restrict the web browser to only connect to particular servers (such as on an intranet) the allowed_ hosts_ to_ connect_to parameter value can be changed.

```
application firefox
{
    binarypaths /usr/bin/firefox:/usr/bin/X11/firefox:
           /usr/lib/firefox/firefox:/usr/lib/firefox/firefox.sh;
    functionality Standard_Graphical_Application
        (peruser_directory="/home/*/.mozilla/firefox/",
        peruser_files="/home/*/.mozilla/appreg",
        application_libraries_directory="/usr/lib/firefox/",
        libraries_fileextension="*.so",
        config_directory={"/home/*/.mozilla/":"/home/*/.gnome2_private/"},
        config_files="",
        read_only_directory="");
    functionality Web_Browser
        (plugins_and_extensions_directory={"/home/*/.mozilla/plugins/":
            "/usr/lib/firefox/extensions/":
            "/usr/lib/browser-plugins/firefox/"},
        download_directory={"/home/*/Desktop/":"/home/*/downloads/"},
        allowed_hosts_to_connect_to="*",
        view_web_files_in_directory="/home/**/");
}
```

Fig. 1. Entire FBAC-LSM policy for Mozilla Firefox

5.2 Defining Functionalities

Each high level functionality is made up of lower level functionalities and privileges. For example, the “Web_Browser” functionality incorporates many inherited functionalities including “http_client”, “Ftp_Client” and “Web_Files_Viewer” which are in turn made up of other functionalities and direct privileges.

The “Web_Browser” functionality policy shown in Figure 2 is syntactically the same as the application policy in the previous figure, with additional concepts such as the definition of parameters (followed by their default values), and information for the graphical tool. Descriptions of functionalities and parameters assist the user, while the granularity of the functionality (high or low level) and a category (in the “Web_ Browser” case “network_client”) allow the graphical tool to flexibly present the policy to the user. Note the scalability of the abstractions provided by the functionalities construct is demonstrated by the fact that the four web browsers considered in the case study use the same underlying functionality definition.

```
functionality Web_Browser
{
      functionality_description "a web browser, and ftp client";
      highlevel;
      category network_client;
      parameter plugins_and_extensions_directory
               "/home/*/.[APPLICATION_NAME]/plugins/";
      param_description "the directory the application keeps any app-specific
      plugins or extensions";
      parameter download_directory "/home/*/downloads";
      param_description "the directories downloads are stored to";
      parameter allowed_hosts_to_connect_to "*";
      param_description "hosts the browser can connect to";
      parameter view_web_files_in_directory "/home/**/";
      param_description "view web files in this dir (.htm, .jpg...)";
      functionality general_network_connectivity_and_file_access ( );
      functionality http_client (allowed_hosts_to_connect_to, <default>);
      functionality save_downloads (download_directory);
      functionality extensions_plugins (plugins_and_extensions_directory, "*");
      functionality mime_aware ( );
      functionality web_plugins_and_helpers ( );
      functionality Ftp_Client (allowed_hosts_to_connect_to);
      functionality Web_Files_Viewer (view_web_files_in_directory, <default>);
}
```

Fig. 2. FBAC-LSM web browser functionality definition

Privileges are low level rights defined as operations on objects, which represent the security-related kernel actions which allow access to the resources that are necessary for that functionality. Low level functionalities, such as "files_r" in Figure 3, provide abstractions to group together related low level privileges. In this functionality the parameter "files" is used to grant both read and "get attribute" access to these files.

```
functionality files_r
{
    functionality_description "read access to these files";
    lowlevel;
    parameter files "";
    param_description "allows these files to be accessed as described";
    privilege file_read files;
    privilege file_getattr files;
}
```

Fig. 3. Low level FBAC-LSM functionality and privileges

The results are policies for these web browsers which enforce the principle of least privilege by confining the application to a restricted set of privileges required for the application to complete its required duties. Consequently the actions of any malware or the effects of any exploited security vulnerability are confined to the behaviour allowed by its functionality-oriented policy.

5.3 Comparison with Other Mechanisms

The FBAC model has significant advantages over existing systems. A policy to confine a complex application such Firefox using standard system call interposition mechanisms such as Systrace or Janus results in a complex series of low level rules specifying which system calls are allowed and under what circumstances. This is illustrated by the excerpt from a Systrace policy given in Figure 4 which only represents a tiny portion of the complete policy. The resulting policy is generally extremely complex and it is difficult to verify that this policy is in fact correct [38].

```
   native-fsread: filename eq "/usr/libexec/ld.so" then permit
   native-fsread: filename eq "/usr/sbin/suexec" then permit
   native-fsread: filename eq "/var/run/ld.so.hints" then permit
   native-fsread: filename eq "/var/www" then permit
   native-fsread: filename eq "<non-existent filename>" then deny[enoent]
```

Fig. 4. Excerpt from a Systrace policy

Similarly, managing NSA's SELinux policy requires expertise beyond that of typical users or system administrators. Under SELinux the policy which applies is the net result of the configuration of multiple access control models (including RBAC, DTE, Multi-level Security and User Identity) and can be hard to verify for correctness or completeness [1, 39]. For example, Figure 5 demonstrates the complexity and inscrutability of a SELinux policy by providing a brief excerpt from an SELinux reference policy for Mozilla [40]. Although domains serve as policy abstractions, each application is usually assigned a unique domain consisting of complex rules specifying allowed file and domain transitions and interactions with types (similarly labelled objects). While SELinux is capable of meeting strong confidentiality requirements, it is not well suited to end users confining potentially malicious applications [41].

```
   manage_dirs_pattern($2,$1_mozilla_home_t,$1_mozilla_home_t)
   manage_files_pattern($2,$1_mozilla_home_t,$1_mozilla_home_t)
   manage_lnk_files_pattern($2,$1_mozilla_home_t,$1_mozilla_home_t)
   relabel_dirs_pattern($2,$1_mozilla_home_t,$1_mozilla_home_t)
   relabel_files_pattern($2,$1_mozilla_home_t,$1_mozilla_home_t)
   relabel_lnk_files_pattern($2,$1_mozilla_home_t,$1_mozilla_home_t)
   manage_files_pattern($1_mozilla_t,$1_mozilla_tmpfs_t,$1_mozilla_tmpfs_t)
   manage_lnk_files_pattern($1_mozilla_t,$1_mozilla_tmpfs_t,$1_mozilla_tmpfs_t)
   manage_fifo_files_pattern($1_mozilla_t,$1_mozilla_tmpfs_t,$1_mozilla_tmpfs_t)
   manage_sock_files_pattern($1_mozilla_t,$1_mozilla_tmpfs_t,$1_mozilla_tmpfs_t)
   fs_tmpfs_filetrans($1_mozilla_t,$1_mozilla_tmpfs_t,{ file lnk_file sock_file
fifo_file })
   allow $1_mozilla_t $2:process signull;
   domain_auto_trans($2, mozilla_exec_t, $1_mozilla_t)
   # Unrestricted inheritance from the caller.
   allow $2 $1_mozilla_t:process { noatsecure siginh rlimitinh };
```

Fig. 5. Excerpt from Mozilla interface rules in the Tresys SELinux reference policy [42]

Novell's AppArmor policy specification format lists the resources an application may access along with the type of access required [14]. This is illustrated in Figure 6. Although this simplifies policy readability, it exposes the underlying complexity of the system. As a result an in-depth knowledge of both the application being confined and low-level details of the operating system's shared resources and services are required in order to properly review the automatically generated policy. Construction of AppArmor policies typically relies on recording process activity, while FBAC policies are constructed based on high level security goals. Further, while AppArmour allows collections of access rules to be grouped into abstractions, these are comparatively inflexible. For example, unlike AppArmour, FBAC has the ability to disable parts

```
    /etc/mailcap r,
    /etc/mime.types r,
    /etc/mozpluggerrc r,
    /etc/opt/gnome/gnome-vfs-*/modules r,
    /etc/opt/gnome/gnome-vfs-*/modules/*.conf r,
    /etc/opt/gnome/pango/* r,
    /etc/opt/kde3/share/applications/mimeinfo.cache r,
    /etc/rpc r,
    /etc/sysconfig/clock r,
    /opt/gnome/lib/GConf/2/gconfd-2 Px,
    /opt/gnome/lib/gnome-vfs-*/modules/*.so mr,
    /opt/gnome/lib/gtk-*/**.so* mr,
    /opt/gnome/lib/lib*so* mr,
    /opt/gnome/lib/pango/**.so mr,
    /opt/gnome/lib64/lib*so* mr,
```

Fig. 6. Excerpt from AppArmor's Firefox profile

of policy on the fly and specify separation of duty, while the parameterised nature of FBAC functionalities allows these to be easily adapted to differing application requirements.

MAPbox provides behaviour based application confinement by allowing software authors to specify a program's behaviour class which describes generally what the program does along with some application-specific parameters [33, 43]. MAPbox's designers identified 14 program classes and corresponding restricted environments are associated with applications based on these author-assigned classes. These restricted environments are defined by complex finely-grained rules specified by the user. While the use of behavioural classes to create an association between policies and programs is an important contribution, policy management in MAPbox remains complex for users. Furthermore applications may only be associated with a single behavioural class which is problematic given many contemporary applications provide a variety of functionality; for example, the Opera web browser. Like MAPbox, FBAC also restricts applications based upon parameterised classes. However, FBAC allows applications to be associated with multiple functionalities and its hierarchical approach to policy management supports multiple levels of abstraction, bringing numerous advantages. For example, FBAC functionalities may be defined hierarchically whereas MAPbox's sandboxes are defined individually. Unlike MAPbox, FBAC allows users to easily restrict arbitrary applications to protect themselves from programs they do not trust. Furthermore FBAC-LSM's use of the LSM interface avoids the problems inherent in MAPbox's use of the system call interface as a security layer [38].

Generally therefore, in contrast to these mechanisms, FBAC-LSM separates and abstracts the task of developing low level policy rules from the task of defining the expected behaviour of a specific program. This allows users, administrators and software authors — in fact uniquely any combination of authorised policy sources — to restrict what an application can do using high level abstractions which can then be easily fine-tuned via parameterisation to suit different applications. Compared to alternative finely-grained application confinement models, FBAC may be used to confine very complex software packages such web browsers using a hierarchical policy that is far easier to manage. While the above confinement methods are either system wide and mandatory (SELinux/DTE, AppArmor) or per-user and discretionary (Janus/Systrace, MAPbox), FBAC-LSM simultaneously enforces mandatory and discretionary FBAC policies. Under FBAC users can configure their own security policy to protect themselves while administrators are able to define system-wide policies to protect system security, enforce organisation level security goals and, when necessary, administer policy to protect specific users. These restrictions on applications severely limit the impact from malware or exploitation of any software vulnerabilities.

6 Discussion

6.1 Manageability and Usability

The policies for the four web browsers presented here (in Figure 1 and Appendix A) are defined in terms of high level security goals. In contrast with other application confinement schemes such as those previously discussed, FBAC allows succinct high level policies to be defined using flexible abstractions. This both reduces and simplifies the management task involved in creating policies to confine individual applications. The programs are simply identified as web browsers and application specific information is supplied. In the case of Opera other high level functionalities are also specified. The finely grained privileges inherited by functionalities are separated from the specification of application policies, thus making policy specification easier than with other schemes. The flexibility to restrict applications based on abstract descriptions of what the application can do provides a significant improvement in usability, making it easier to translate high level security requirements into finely grained policies.

Furthermore, the “Web_Browser” functionality (in Figure 2) demonstrates that the hierarchical structure of policies allows functionalities themselves to also be defined in terms of abstractions, such as “http_client”. Policies can be reviewed from their high level functionalities (such as “Web_Browser” in Figure 2) to lower level detail and right down to the privileges specifying permissible operations on designated objects (such as “file_r” in Figure 3). This makes finely grained policies easier to manage and comprehend as the policy is made up of levels of abstractions which can encapsulate low-level details.

6.2 Scalability

Once functionalities such as “Web_Browser” have been defined, policies for each application which provides the described functionality can be defined in terms of these constructs. All four web browsers studied reuse the “Web_Browser” policy abstraction. This leverages the fact that many applications may be categorised into the same behavioural classes and can be confined to easily identified sets of privileges required for the applications to carry out their intended functions [33]. Rather than confining an application by specifying each distinct privilege required, they can be simply defined in terms of the behavioural classes to which they belong. Thus the model scales well to confine the numerous applications typically found on contemporary systems.

The use of functionality hierarchies also increases the scalability of policy management by facilitating greater reuse of existing defined policy. For example the “Web_Browser” functionality includes the functionality “Ftp_Client” which itself can be used to describe applications which may not be web browsers. The use of hierarchies increases abstraction while reducing redundancy.

6.3 Security

Using application confinement schemes such as FBAC to limit program privilege provides significant security improvements over simply relying on user-oriented access control mechanisms. FBAC enforces the principle of least privilege by confining applications to the set of privileges required for them to do their job. Although the abstract nature of functionalities may potentially grant an application more privileges than they actually use, in general these additional privileges simply allow the application to carry out its authorised tasks in varied ways.

If an application attempts to exercise privileges it does not hold the request is denied. For example, if due to the introduction of malicious code a restricted web browser attempts to act outside of the behaviour defined by its associated functionalities the action would be prevented. This limits the ability of applications to behave maliciously whether deliberately or otherwise.

The underlying FBAC-LSM policy granularity is finely grained and is determined by the LSM interface. This design provides scope for the future inclusion of additional features such as stateful network packet inspection.

Compared with other confinement models the FBAC framework provides equivalent security benefits. However, the superior convenience, simplicity, flexibility and scalability of the FBAC model makes it far better suited to ubiquitous deployment.

However, beyond this FBAC has other security advantages. For example, the separation of duty feature is unique in the area of application confinement and allows high level security policies to specify privileges or functionalities that cannot be exercised simultaneously. Static separation of duty prevents conflicting privileges from being assigned to the same application while dynamic separation of duty stops applications from exercising certain privileges concurrently. This limits the ability of high level security goals to be accidentally subverted by low-level security policies.

Also, as FBAC's policy abstractions are natively hierarchical, parts of the policy can be easily activated or deactivated at run time. This is not possible using the existing application-oriented access control models such as DTE, RC or AppArmor as privileges are contained in a monolithic abstraction associated with the security context. FBAC's hierarchy of functionalities allows run-time intervention to dynamically deactivate or activate branches of functionalities. This could be requested by a user, administrator or the software itself. For example using a multi-purpose application (such as Opera web browser, email, irc, new reader and bittorrent client) the user or the application itself may wish to only enable the functionality corresponding to the feature the program is performing. This is equivalent to the concept of an RBAC user activating only those roles corresponding to the job he or she is currently performing.

FBAC also restricts applications based on a combination of policies representing the security goals of users and administrators. While existing controls provide either mandatory or discretionary application confinement, FBAC provides both through layers of confinements. Policy can be reused across confinements and only one mechanism needs to be maintained.

7 Conclusion

The case study and corresponding analysis of the FBAC model presented here demonstrates that applying parameterised RBAC constructs to the problem of application confinement can provide clear advantages over alternative approaches. FBAC separates the task of policy construction from the association of these policies with specific applications. This simplifies the process as users or administrators can assign pre-specified generic policies based upon an application's anticipated functionality rather than needing to construct individual policies for each program. FBAC utilises functionalities as an abstract policy construct and by allowing the definition of new functionalities in terms of existing ones, a hierarchy is created which improves usability, manageability and scalability. While end users can simply assign policies based upon high-level functionalities, security administrators and analysts can study policy construction at multiple levels. The separation of duty mechanism also ensures that high level policy goals are maintained during the construction of low-level policies. Finally, the use of parameterisation allows confinement policies to be easily adapted to deal with subtle differences between similar applications and this further improves policy reusability. As demonstrated by the four web browser policies presented, the usability and management improvements provided by FBAC make deploying application confinement significantly easier and could therefore have the potential to encourage broader adoption of such security mechanisms in the future.

References

1. Zanin, G., Mancini, L.V.: Towards a Formal Model for Security Policies Specification and Validation in the SElinux System. In: Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies, pp. 136–145. ACM Press, Yorktown Heights (2004)
2. Goldberg, I., Wagner, D., Thomas, R., Brewer, E.A.: A Secure Environment for Untrusted Helper Applications: Confining the Wily Hacker. In: Proceedings of the 6th USENIX Security Symposium. University of California, San Jose (1996)
3. Kamp, P.-H., Watson, R.: Building Systems to be Shared Securely. ACM Queue 2, 42–51 (2004)
4. Madnick, S.E., Donovan, J.J.: Application and Analysis of the Virtual Machine Approach to Information Security. In: Proceedings of the ACM Workshop on Virtual Computer Systems, Cambridge, MA, USA, March 1973, pp. 210–224 (1973)
5. Kamp, P.-H., Watson, R.: Jails: Confining the Omnipotent Root. In: Sane 2000 - 2nd International SANE Conference (2000)
6. Tucker, A., Comay, D.: Solaris Zones: Operating System Support for Server Consolidation. In: 3rd Virtual Machine Research and Technology Symposium Works-in-Progress
7. Whitaker, A., Shaw, M., Gribble, S.D.: Lightweight virtual machines for distributed and networked applications. In: Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation, pp. 195–209 (2002)
8. Gong, L., Mueller, M., Prafullchandra, H., Schemers, R.: Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2. In: USENIX Symposium on Internet Technologies and Systems. Prentice Hall PTR, Monterey (1997)

9. Thorsteinson, P., Ganesh, G.G.A.: Net Security and Cryptography, p. 229. Prentice Hall PTR, Englewood Cliffs (2003)
10. Li, N., Mao, Z., Chen, H.: Usable Mandatory Integrity Protection for Operating Systems. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 164–178 (2007)
11. Sun, W., Sekar, R., Poothia, G., Karandikar, T.: Practical Proactive Integrity Preservation: A Basis for Malware Defense. Security and Privacy. In: IEEE Symposium on SP 2008, pp. 248–262 (2008)
12. Wagner, D.A.: Janus: An Approach for Confinement of Untrusted Applications. Technical Report: CSD-99-1056. Electrical Engineering and Computer Sciences. University of California, Berkeley, USA (1999)
13. Provos, N.: Improving Host Security with System Call Policies. In: 12th USENIX Security Symposium, vol. 10. USENIX, Washington (2002)
14. Cowan, C., Beattie, S., Kroah-Hartman, G., Pu, C., Wagle, P., Gligor, V.: SubDomain: Parsimonious Server Security. In: USENIX 14th Systems Administration Conference (LISA) (2000)
15. Berman, A., Bourassa, V., Selberg, E.: TRON: Process-Specific File Protection for the UNIX Operating System. In: Proceedings of the 1995 Winter USENIX Conference (1995)
16. Bacarella, M.: Taking advantage of Linux capabilities. Linux Journal (2002)
17. Krsti, I., Garfinkel, S.L.: Bitfrost: the one laptop per child security model. In: ACM International Conference Proceeding Series, vol. 229, pp. 132–142 (2007)
18. Miller, M.S., Tulloh, B., Shapiro, J.S.: The structure of authority: Why security is not a separable concern. In: Multiparadigm Programming in Mozart/Oz: Proceedings of MOZ 3389 (2004)
19. Stiegler, M., Karp, A.H., Yee, K.P., Close, T., Miller, M.S.: Polaris: virus-safe computing for Windows XP. Communications of the ACM 49, 83–88 (2006)
20. Wagner, D.: Object capabilities for security. In: Conference on Programming Language Design and Implementation: Proceedings of the 2006 workshop on Programming languages and analysis for security, vol. 10, pp. 1–2 (2006)
21. Badger, L., Sterne, D.F., Sherman, D.L., Walker, K.M., Haghighat, S.A.: Practical Domain and Type Enforcement for UNIX. In: Proceedings of the 1995 IEEE Symposium on Security and Privacy, p. 66. IEEE Computer Society, Los Alamitos (1995)
22. Ott, A.: The Role Compatibility Security Model. In: 7th Nordic Workshop on Secure IT Systems (2002)
23. Krohn, M., Efstathopoulos, P., Frey, C., Kaashoek, F., Kohler, E., Mazieres, D., Morris, R., Osborne, M., VanDeBogart, S., Ziegler, D.: Make least privilege a right (not a privilege). In: Procedings of 10th Hot Topics in Operating Systems Symposium (HotOS-X), Santa Fe, NM, USA, pp. 1–11 (2005)
24. Marceau, C., Joyce, R.: Empirical Privilege Profiling. In: Proceedings of the 2005 Workshop on New Security Paradigms, pp. 111–118 (2005)
25. Jaeger, T., Sailer, R., Zhang, X.: Analyzing Integrity Protection in the SELinux Example Policy. In: Proceedings of the 12th USENIX Security Symposium, pp. 59–74 (2003)
26. Hinrichs, S., Naldurg, P.: Attack-based Domain Transition Analysis. In: 2nd Annual Security Enhanced Linux Symposium, Baltimore, Md., USA (2006)
27. Ferraiolo, D., Kuhn, R.: Role-Based Access Control. In: 15th National Computer Security Conference, Baltimore, MD, USA, pp. 554–563 (1992)
28. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-Based Access Control Models. IEEE Computer 29, 38–47 (1995)
29. Simon, R.T., Zurko, M.E.: Separation of Duty in Role-Based Environments. In: Proceedings of 10th IEEE Computer Security Foundations Workshop, Rockport, MD, pp. 183–194 (1997)

30. Schreuders, Z.C., Payne, C.: Functionality-Based Application Confinement: Parameterised Hierarchical Application Restrictions. In: Proceedings of SECRYPT 2008: International Conference on Security and Cryptography, pp. 72–77. INSTICC Press, Porto (2008)
31. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security 4, 224–274 (2001)
32. ANSI INCITS 359-2004. American National Standards Institute / International Committee for Information Technology Standards (ANSI/INCITS)
33. Acharya, A., Raje, M.: MAPbox: Using Parameterized Behavior Classes to Confine Applications. In: Proceedings of the 2000 USENIX Security Symposium, Denver, CO, USA (2000)
34. Jaeger, T., Prakash, A.: Requirements of role-based access control for collaborative systems. In: Proceedings of the first ACM Workshop on Role-based access control, p. 16. ACM Press, Gaithersburg (1996)
35. Friberg, C., Held, A.: Support for discretionary role based access control in ACL-oriented operating systems. In: Proceedings of the second ACM workshop on Role-based access control, pp. 83–94. ACM Press, Fairfax (1997)
36. Jansen, W.A.: Inheritance Properties of Role Hierarchies. In: Proceedings of the 21st National Information Systems Security Conference, pp. 476–485. National Institute of Standards and Technology, Gaithersburg (1998)
37. Wright, C., Cowan, C., Smalley, S., Morris, J., Kroah-Hartman, G.: Linux Security Module Framework. In: Ottawa Linux Symposium, Ottawa, Canada (2002)
38. Garfinkel, T.: Traps and Pitfalls: Practical Problems in System Call Interposition Based Security Tools. In: Proceedings of the 10th Network and Distributed System Security Symposium, pp. 163–176. Stanford University, San Diego (2003)
39. Bratus, S., Ferguson, A., McIlroy, D., Smith, S.: Pastures: Towards Usable Security Policy Engineering. In: Proceedings of the Second International Conference on Availability, Reliability and Security, pp. 1052–1059 (2007)
40. Tresys: SELinux Reference Policy (2008)
41. Harada, T., Horie, T., Tanaka, K.: Towards a manageable Linux security. In: Linux Conference 2005 (Japanese) (2005), http://lc.linux.or.jp/lc2005/02.html
42. Tresys: SELinux Reference Policy (2008), http://oss.tresys.com/projects/refpolicy
43. Raje, M.: Behavior-based Confinement of Untrusted Applications. TRCS 99-12. Department of Computer Science. University of Calfornia, Santa Barbara (1999)

Appendix A: FBAC-LSM Policies for Popular Web Browsers

```
application lynx
{
    binarypaths /usr/bin/lynx:/usr/bin/X11/lynx;
    functionality Standard_Commandline_Application
        (peruser_directory="",
        peruser_files="",
        application_so_libraries_directory="",
        libraries_fileextension="",
        config_directory="",
        config_files={"/etc/lynx.cfg":"/etc/lynx.lss"},
        read_only_config_directory="");
    functionality Web_Browser
        (plugins_and_extensions_directory="",
        download_directory="/home/*/downloads/",
        allowed_hosts_to_connect_to="*",
        view_web_files_in_directory="/home/**/");
    functionality user_login_awareness ( );
    functionality requires_tmp_access ( );
}
```

```
application epiphany
{
    binarypaths /usr/bin/epiphany:/usr/bin/X11/epiphany;
    functionality Standard_Graphical_Application
        (peruser_directory="/home/*/.gnome2/epiphany/",
        peruser_files="/home/*/.gnome2/accels/epiphany",
        application_libraries_directory="/usr/lib/epiphany/",
        libraries_fileextension="*",
        config_directory="/home/*/.gnome2_private/",
        config_files={"/home/*/.mozilla/firefox/profiles.ini":
            "/home/*/.mozilla/firefox/*/prefs.js"},
        read_only_directory="/usr/share/epiphany/");
    functionality Web_Browser
        (plugins_and_extensions_directory="/usr/share/epiphany-extensions/",
        download_directory="/home/*/downloads/",
        allowed_hosts_to_connect_to="*",
        view_web_files_in_directory="/home/**/");
    functionality register_as_mozplugger_plugin ( );
}

application opera
{
    binarypaths /usr/bin/opera:/usr/bin/X11/opera;
    functionality Standard_Graphical_Application
        (peruser_directory="/home/*/.opera/",
        peruser_files="",
        application_libraries_directory="/usr/lib/opera/",
        libraries_fileextension="*",
        config_directory="/home/*/.kde/share/config/",
        config_files={"/etc/opera6rc":"/etc/opera6rc.fixed"},
        read_only_directory="/usr/share/opera/");
    functionality Web_Browser
        (plugins_and_extensions_directory={"/usr/lib/opera/plugins/":
        "/usr/lib/browser-plugins/":"/usr/lib/firefox/plugins/"},
        download_directory={"/home/*/OperaDownloads/":
        "/home/*/downloads/"},
        allowed_hosts_to_connect_to="*",
        view_web_files_in_directory="/home/**/");
    functionality Email_Client
        (mail_out_SMTP_servers="my.mail.server.com",
        SMTP_remote_port=<default>,
        mail_in_POP3_servers="*",
        POP3_remote_port=<default>,
        mail_in_IMAP_servers="*",
        IMAP_remote_port=<default>);
    functionality Irc_Chat_Client
        (chat_IRC_servers=<default>,
        IRC_remote_port=<default>);
    functionality News_Reader_Client (news_NNTP_servers=<default>);
    functionality BitTorrent_Client
        (bittorrent_peers_and_trackers="*",
        bittorrent_remote_port="18768");
}
```

Towards Role Based Trust Management without Distributed Searching of Credentials

Gang Yin[1], Huaimin Wang[1], Jianquan Ouyang[2], Ning Zhou[3], and Dianxi Shi[1]

[1] School of Computer, National University of Defense Technology, Changsha, China
jack.nudt@gmail.com, whm_w@163.com, dxshi@nudt.edu.cn
[2] College of Information Engineering, Xiangtan University, Xiangtan, China
oyjq@ict.ac.cn
[3] Institute of Electronic System Engineering of China, Beijing, China
humimi74@yahoo.cn

Abstract. Trust management systems enable decentralized authorization by searching distributed credentials from network. We argue that such distributed searching processes may encounter many technical or non-technical problems, and can be avoided by storing delegation credentials redundantly with acceptable costs. We propose a scoped-role based trust management system ScoRT, using a novel credential affiliation model to compute the credentials necessary for role membership decisions, which can be used to guide the storage, retrieval and revocation of credentials. The algorithm for distributed credential storage and retrieval is designed based on the model and its sound and complete properties are formally analyzed with respect to ScoRT semantics. Complexity analysis and estimation show that, by redundantly storing acceptable amount of delegation credentials, ScoRT enables more practical and automatic authorization without searching credentials from remote entities, and thus helps to overcome the deficiencies of existing approaches.

1 Introduction

Trust management (TM) systems use delegation credentials to realize flexible and scalable authorization across security domains. A number of TM systems have been proposed to enable various delegation mechanisms, such as [2, 4, 5, 7, 10, 14, 15, 16]. The primary idea of delegation is that one entity gives some of its authority to others to make authorizations on behalf of the former. Multi-steps of delegation among different entities may result in chains of credentials, which are prerequisite for making authorization decisions in TM systems. Some TM systems study the distributed storage and retrieval problems of credential chains, which can be mainly classified into logic-based approach [1, 12, 14] and graph-based approach [13, 16, 18].

The logic-based approach requires that each credential is defined with a specified location, and servers pull credentials from remote entities (usually one credential at a time) during the process of logic-based compliance-checking, such as QCM [12], SD3 [14] and Cassandra [2]. The graph-based approach retrieves the whole credential chains that delegate the privileges from authorizers to requesters, such as the discovery methods of certificate paths or credential chains [8, 9, 13, 16]. These two approaches

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 222–237, 2008.

are mainly used to search credentials in dynamic and decentralized environments, which may face a lot of technical and non-technical problems. (1) They use the depth-first or breadth-first search process to find credential chains, which will retrieve a lot of useless credentials during back-tracking processes, and if the authorization queries have negative results, the credentials may become tremendous. (2) The issuers of the target credentials may not always trust or be trusted by the credential requesters, and thus the privacy of credentials may be breached by uncontrolled credential retrieval. (3) Entities holding the target credentials may not on-line all the time and thus the requests will be rejected if any credential in the chain can not be retrieved in time. These problems may breach the security and availability of systems.

The primary task of credential searching is to retrieve delegation credentials from remote entities. Above observations motivate us to revisit the problem of distributed management of credentials. Due to the balance of scalability and controllability of delegation mechanisms, we believe that delegation credentials should be used and configured as the backbone of collaboration networks. For example, M. Becker gives a comprehensive analysis of policies in EHR systems [3], the delegation credentials across security domains are only 8% of the total 375 credentials.

In this paper, we propose a novel credential distribution approach to store delegation credentials redundantly such that every entity can hold all necessary delegation credentials for making access decisions. To ensure the generality of our approach, we propose ScoRT, a role-based TM system which combines the primary capabilities of both RT [16] and SPKI [10]. The credential affiliation model is proposed to computes the affiliation graphs for roles, and each affiliation graph contains all the delegation chains starting from these roles. A credential distribution algorithm is designed based on the model to publish and retrieve the credentials among the entities in the network.

The rest of this paper is organized as follows. Section 2 defines ScoRT and the credential affiliation model. Section 3 introduces the credential management framework of ScoRT and proves the sound and complete properties of credential distribution algorithm. Section 4 estimates the complexity of our approach based on Aura's delegation network model. Section 5 analyzes the related works and section 6 concludes the paper.

2 Credential Affiliation in Trust Management

ScoRT is a role-based TM system combing the primary advantages of both role-based trust management [17] and SPKI [10]. ScoRT uses scoped roles to enable and control the delegation of role-based privileges.

2.1 ScoRT: Trust Management with Scoped-Roles

ScoRT introduces the notion of scoped roles, and a scoped role is a role appended with a trust scope tag, which is either $\blacksquare$ or $\square$. Given an entity A and a role name r, the $A.r$ has two scoped roles $A.r.\blacksquare$ and $A.r.\square$. The scoped roles can be regarded as a kind of refined abstraction of principals. Intuitively, $A.r.\blacksquare$ denotes the entities directly assigned with $A.r$ by A, while $A.r.\square$ denotes the entities directly or indirectly assigned with $A.r$. ScoRT has three kinds of credentials:

- type-1: $A.r{\leftarrow}B$
- Entity B is a member of the role $A.r$.
- type-2: $A.r{\leftarrow}B.r_1.s$
- Members of the scoped role $B.r_1.s$ are members of the role $A.r$.
- type-3: $A.r{\leftarrow}B_1.r_1.s_1{\wedge}B_2.r_2.s_2$
- Members of both $B_1.r_1.s_1$ and $B_2.r_2.s_2$ are members of the role $A.r$. We call $B_1.r_1.s_1{\wedge}B_2.r_2.s_2$ a scoped role intersection or an intersection for short.

The type-1 credentials are authorization credentials, while the type-2 and type-3 credentials are delegation credentials. Given a credential c, the entity at left-side is called the issuer of c, and the entity at right-side is called subject of c. ScoRT only supports the intersections of two scoped roles, but the intersections of more scoped roles can be defined by introducing new intermediate credentials. For example, the intersection of $B_1.r_1.s_1$ and $B_2.r_2.s_2$ can be replaced by $C.r.\square$ by introducing the credential $C.r{\leftarrow}B_1.r_1.s_1{\wedge}B_2.r_2.s_2$.

Example 1. The following ScoRT credentials have similar meanings to the sample credentials in [16], but these credentials can provide more refined delegation control.

(1) EPub.discount ← EOrg.preferred.□ ∧ ACM.member.■
(2) EOrg.preferred ← StateU.student.□ (3) StateU.student ← RegB.student.■
(4) ACM.member ← Alice (5) RegB.student← Alice

The credential (3) means that all students *directly* assigned by RegB can be the students of StateU, which *implicitly* defines the delegation from StateU to RegB of StateU.student within the depth 2. The credential (2) defines delegation from EOrg to StateU of EOrg.preferred without depth control. The semantics of ScoRT is defined by Datalog [19], which decides whether an entity is a member of a given role.

Definition 1 (Semantics of ScoRT). Given a credential set Σ, a role $A.r$ and an entity B, we use $\Sigma \hookrightarrow \text{mem}(B, A.r)$ to denote that B is a member of $A.r$, and:

$$\Sigma \hookrightarrow \text{mem}(B, A.r) \text{ iff } \mathrm{P}_\Sigma \vdash \text{m}(B, A.r, \square)$$

where P_Σ is the set of definite Datalog rules derived from Σ, $\vdash$ is the logical consequence relation. P_Σ contains a rule:

$$\text{m}(x, y.r, \square) \leftarrow \text{m}(x, y.r, \blacksquare). \quad \text{(R1)}$$

P_Σ also contains the rules obtained by a transform process. For each $A.r{\leftarrow}b{\in}\Sigma$, do the transform according to three rules:
(T1) if b is an entity, then $\text{m}(b, A.r, \blacksquare){\in}\mathrm{P}_\Sigma$;
(T2) if b is $B.r_1.s$ then $\text{m}(x, A.r, \square) \leftarrow \text{m}(x, B.r_1, s){\in}\mathrm{P}_\Sigma$;
(T3) if b is $B_1.r_1.s_1{\wedge}B_2.r_2.s_2$ then $\text{m}(x, A.r, \square) \leftarrow \text{m}(x, B_1.r_1, s_1), \text{m}(x, B_2.r_2, s_2){\in}\mathrm{P}_\Sigma$.

2.2 Credential Affiliation

The credential affiliation model is built upon ScoRT. Given a set Σ of ScoRT credentials, the authorization structures of Σ can be modeled by a partial weighed directed graph, in which the type-1 credentials are mapped to simple directed edges, while the type-2 and type-3 are mapped to weighed directed edges.

Given a set Σ of credentials, we use $\Sigma.\mathrm{entities}$, $\Sigma.\mathrm{roles}$ and $\Sigma.\mathrm{scoints}$ to denote the set of all entities appeared in the right side of type-1 credentials in Σ, the set of all roles in the credentials in Σ, and all the intersections appeared in Σ.

Definition 2 (Credential Graph). Given a set Σ of credentials, the credential graph for Σ is denoted by a weighed directed graph G_Σ which has a node set N_Σ and an edge set E_Σ, defined as follows:

$$N_\Sigma = \Sigma.\mathrm{entities} \cup \Sigma.\mathrm{roles} \cup \Sigma.\mathrm{scoints}$$
$$E_\Sigma \subseteq E_\Sigma.\mathrm{AE} \cup E_\Sigma.\mathrm{TE} \cup E_\Sigma.\mathrm{IE}$$

where $E_\Sigma.\mathrm{AE}$, $E_\Sigma.\mathrm{TE}$ and $E_\Sigma.\mathrm{IE}$ are the set of authorization edges, trust edges and intersection edges in E_Σ, which are further defined as follows:

$$E_\Sigma.\mathrm{AE} \subseteq \Sigma.\mathrm{roles} \times \Sigma.\mathrm{entities}$$
$$E_\Sigma.\mathrm{TE} \subseteq (\Sigma.\mathrm{roles} \cup \Sigma.\mathrm{scoints}) \times \Sigma.\mathrm{roles} \times \mathrm{TS}$$
$$E_\Sigma.\mathrm{IE} \subseteq \Sigma.\mathrm{roles} \times \Sigma.\mathrm{scoints} \times \wp(\Sigma.\mathrm{entities})$$

where TS={■, □} and $\wp(\Sigma.\mathrm{entities})$ is the power set of $\Sigma.\mathrm{entities}$. G_Σ uses a long arrow $\longleftarrow$ to denote a directed edge and a weighed long arrow $\overset{w}{\longleftarrow}$ to denote a weighed directed edge with the weight w. If there is an edge from n to n', then n' is a *successor* of n. Given a path ξ from node n to n' in G_Σ, $n \neq n'$ and n'' is the successor of n in ξ, then ξ is a legal path denoted by $n' \twoheadleftarrow n$ if one of the following is satisfied:

- if n is a role, then the weight of each trust edge between n'' and n' in ξ is □.
- if n is an entity, then the sub-path from n'' to n' in ξ is a legal path and the weight of each intersection edge in the sub-path contains n.
- if ξ is a sub-path of a legal path, then ξ is a legal path.

We use $n' \twoheadleftarrow n \in G_\Sigma$ to denote that $n' \twoheadleftarrow n$ is a legal path in G_Σ. Given $n'' \overset{w}{\longleftarrow} n'$ and $n' \twoheadleftarrow n$ in G_Σ, if the path formed by linking $n'' \overset{w}{\longleftarrow} n'$ and $n' \twoheadleftarrow n$ is a legal path then $n'' \overset{w}{\longleftarrow} n' \twoheadleftarrow n \in G_\Sigma$. Similarly, $n'' \twoheadleftarrow n' \overset{w}{\longleftarrow} n \in G_\Sigma$ if the path formed by linking $n'' \twoheadleftarrow n'$ and $n' \overset{w}{\longleftarrow} n$ is a legal path. The subsets of E_Σ can be constructed by the closure properties:

Closure Property 1: Given $B \in \Sigma.\mathrm{entities}$, $A.r \leftarrow B \in \Sigma$, then $A.r \longleftarrow B \in E_\Sigma.\mathrm{AE}$.

Closure Property 2: Given $A.r \leftarrow b \in \Sigma$ and b is not an entity, then for each scoped role $n.s$ in b, $A.r \overset{s}{\longleftarrow} n \in E_\Sigma.\mathrm{TE}$.

Closure Property 3: Given $A.r \leftarrow b \in \Sigma$ and $b \in \Sigma.\mathrm{scoints}$, if there is no edge from b to $A.r$ then $A.r \overset{\varnothing}{\longleftarrow} b \in E_\Sigma.\mathrm{IE}$.

Closure Property 4: Given $B \in \Sigma.\mathrm{entities}$, $A.r \leftarrow b \in \Sigma$ and $b \in \Sigma.\mathrm{scoints}$, if $A.r \overset{w}{\longleftarrow} b \in E_\Sigma.\mathrm{IE}$ and $b \overset{w}{\longleftarrow} n \twoheadleftarrow B \in G_\Sigma$ for each role n in b then $A.r \overset{w \cup \{B\}}{\longleftarrow} b \in E_\Sigma.\mathrm{IE}$.

The credential graph for the credentials in example 1 is shown in Figure 1. The directed edge ACM.member $\longleftarrow$ Alice is an authorization edge of the credential (4). The weighed directed edges EOrg.perferred $\overset{□}{\longleftarrow}$ StateU.student is a trust edge of the credential (2). Let α be the intersection EOrg.preferred.□∧ACM.member.■, the trust edges

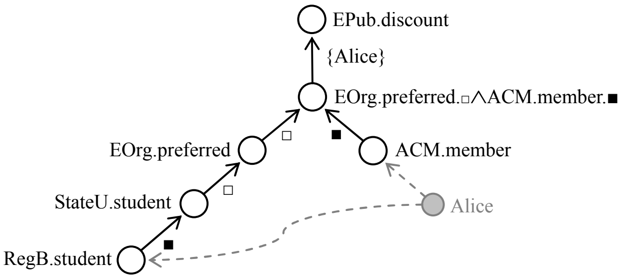


Fig. 1. Weighed Credential Graphs of ScoRT

$\alpha \xleftarrow{\square}$ EOrg.preferred and $\alpha \xleftarrow{\blacksquare}$ ACM.member are derived from credential (1). By definition 2, EOrg.perferred$\twoheadleftarrow$Alice and ACM.member$\twoheadleftarrow$Alice, and thus the edge EPub.discount $\xleftarrow{\{\text{Alice}\}} \alpha$ is the intersection edge of the credential (1).

Given a credential $A.r \leftarrow b$, we use Edges($A.r \leftarrow b$) to denote the edge set S, where S is $\{A.r \longleftarrow b\}$ if b is an entity, and S is $\{A.r \xleftarrow{s} B.r_1\}$ if b is a scoped role $B.r_1.s$, and S is $\{A.r \xleftarrow{\varnothing} b,\ b \xleftarrow{s_1} B_1.r_1,\ b \xleftarrow{s_2} B_2.r_2\}$ if b is an intersection $B_1.r_1.s_1 \wedge B_2.r_2.s_2$. We use the concept of affiliation graph to model the credentials which influence the decision on whether an entity is a member of a specified role.

Definition 3 (Affiliation Graph). Given a set Σ of credentials, a role or intersection n, the affiliation graph of n is denoted by G_Σ^n, and n is called the *root* of G_Σ^n. N_Σ^n and E_Σ^n are sets of nodes and edges in G_Σ^n, constructed by AffG (n, $N_\Sigma^n = \varnothing$, $E_\Sigma^n = \varnothing$):
AffG (n /*input node*/, ns /*node set*/, es /*edge set*/)

```
if $n$ is marked then return;
add $n$ into $ns$; mark $n$ to be processed;
for each edge $e$ of the form $n \xleftarrow{w} n'$ in $G_\Sigma$ do
  add $e$ into $es$;
  if $w$=□ then AffG ($n'$, $ns$, $es$);
  if $w=\varnothing$ then for each scoped role $n''.s$ in $n'$ do
    add $n' \xleftarrow{s} n''$ into $es$;
    if $s$=□ then AffG ($n''$, $ns$, $es$);
return;
```

Both space and time complexities of affiliation graphs are liner to the size of given credentials. By definition 3, type-1 credentials are not used when constructing affiliation graphs. Given a set Σ of non-type-1 credentials, N is the number of credentials in Σ, by definition 2, G_Σ has at most $4N$ nodes ($3N$ role nodes and N intersection nodes) and $4N$ edges ($3N$ trust edges and N intersection edges). Given a role $A.r$, $G_\Sigma^{A.r}$ is a sub-graph of G_Σ by definition 3. Therefore the space complexity of $G_\Sigma^{A.r}$ is $O(N)$.

By definition 2, constructing G_Σ only need one iteration step and time complexity of constructing G_Σ is $O(N)$. By definition 3, every processed node is marked and there are at most 4N nodes in G_Σ, and thus lines 2, 4, 7 of AffG will be executed at most 4N times when constructing $G_\Sigma^{A.r}$. Therefore the time complexity of AffG is O(N).

Lemma 1. Given a credential set Σ, a role $A.r$ and an entity D, then $\Sigma \hookrightarrow \text{mem}(D, A.r)$ if and only if $A.r \twoheadleftarrow D \in G_\Sigma$.

Given a set Σ of credentials, we use $G_\Sigma^{A.r:B}$ to denote an extension of $G_\Sigma^{A.r}$ which is called the specified affiliation graph of $A.r$ for B. The node set and edge set of $G_\Sigma^{A.r:B}$ are $N_\Sigma^{A.r:B} = N_\Sigma^{A.r} \cup \{B\}$ and $E_\Sigma^{A.r:B} = E_\Sigma^{A.r} \cup \Delta_B$, where Δ_B is $\{e \mid e \text{ is } n \longleftarrow B \text{ and } e \in E_\Sigma.\}$ containing all the authorization edges from B. The intuitionistic meaning of $G_\Sigma^{A.r:B}$ is that it contains all the credentials that maybe used to decide whether B is a member of $A.r$. The following lemmas show some basic properties of affiliation graphs.

Lemma 2. Given a set Σ of credentials, let $A.r$ be a role in Σ.roles and n be an node in G_Σ, if $A.r \twoheadleftarrow n \in G_\Sigma$ then $A.r \twoheadleftarrow n \in G_\Sigma^{A.r}$.

Lemma 3. Given a set Σ of credentials, a role $A.r$ and an entity D, then $A.r \twoheadleftarrow D \in G_\Sigma$ if and only if $A.r \twoheadleftarrow D \in G_\Sigma^{A.r:D}$.

From lemma 1 and lemma 3, the affiliation graph is sound and complete with respect to the semantics of ScoRT.

Theorem 1 (Soundness and Completeness of Affiliation Graph). Given a set of Σ of credentials, a role $A.r$ and an entity D, then $\Sigma \hookrightarrow \text{mem}(D, A.r)$ if and only if $A.r \twoheadleftarrow D \in G_\Sigma^{A.r:D}$.

3 Credential Management Framework

ScoRT provides a framework for distributed storage, retrieval and revocation, which is mainly guided by the affiliation graph model. The framework uses different policies to handle authorization credentials and delegation credentials.

ScoRT uses credential distribution algorithm (CDA) to exchange credentials among collaborating entities. When a CDA algorithm is invoked, a CDA instance will be created by its local entity. Given a CDA instance α, if α is invoked by its local entity, then α is a *root* instance, otherwise α is called a *derived* instance ScoRT manages credentials based on the following policies.

Definition 4 (Credential Management Policies). ScoRT manages the credentials according to four general and intuitionistic policies:

Policy CMP1: Authorization credentials are stored at its issuers and subjects, and should be pushed to authorizers if required by authorization checking.

Policy CMP2: Delegation credentials are stored at their issuers and subjects; the affiliation graphs of each role should be stored at its defining entity.

Policy CMP3: Credentials can only be revoked by their issuers by sending revoking messages to related entities, or by setting expiration time for each credentials.

Policy CMP4: Root CDA instances run one by one serially in order to ensure the consistency of credential distribution.

Given a time τ and a credential set Σ, we use $\Sigma|_\tau$ and $G_\Sigma|_\tau$ to denote the snapshots of Σ and G_Σ at τ respectively. Given an entity A, we use Σ_A to denote the credentials stored at A, and use G_A to briefly denote G_{Σ_A}. Given a CDA instance α, we use α^- and α^+ to denote the moments when α just begins and ends.

3.1 Credential Distribution Algorithm

CDA exchanges the credentials among entities based on CMP1 and CMP2. When an entity A issues a credential $A.r \leftarrow b$, it creatse a root CDA instance by calling CDA($A.r \leftarrow b$, *nil*, *nil*). A root CDA instance may retrieve credentials from remote entities (lines 7 and 11), push credentials to remote entities (lines 4, 8 and 12), and invoke derived instances by calling CDA on remote entities (lines 16 and 17).

The statement "*n* contains *v*" in line 14 means that *v* is a role appearing in *n*. Given an entity *B*, we use *B.cda* to denote the invocation of CDA on entity *B*, and thus a chain of CDA instances may be created with cascaded invocations. Given an entity *B* and a credential *c*, the statement "send *c* to *B*" means that after receiving *c*, *B* will store *c* at its local repository and add Edges(*c*) into G_B.

Pseudo-codes of Credential Distribution Algorithm

```
CDA ($A.r \leftarrow$b /*input credential*/, g /*affiliation graph*/, v /*tracing role*/)
1.  let $c$ be $A.r \leftarrow b$; let $es$ be Edges($c$);
2.  add $es$ and $g$ into $G_{ld}$; /*ld is the identity of local entity*/
3.  if $A = ld$ then
4.    if $b$ is an entity $D$ and $b = ld$ then
5.      send $c$ to $D$;
6.    if $b$ is a scoped role $B.r_1.\square$ and $B = ld$ then
7.      pull $G_B^{B.r_1}$ from $B$;
8.      send $c$ to $B$;
9.    if $b$ is a scoped role intersection $B_1.r_1.s_1 \wedge B_2.r_2.s_2$ then
10.     for each $i$ in $\{1, 2\}$ do if $s_i = \square$ and $B_i = ld$ then
11.         pull $G_{B_i}^{B_i.r_i}$ from $B_i$;
12.         send $c$ to $B_i$;
13. for each edge $A'.r' \xleftarrow{w} n$ in $G_{ld}$ do
14.   if $n$ contains $v$ and $A' = ld$ then
15.     if ($w = \square$) or ($w = \varnothing$ and $n \xleftarrow{\square} v \in G_{ld}$) then
16.       if $A = ld$ then call $B.cda(c, G_{ld}^n, A'.r')$; /*calls remote CDA*/
17.       else call $B.cda(c, g, A'.r')$; /*calls remote CDA*/
18. return;
```

CDA stores each credential at its issuers and subjects, because subjects must know their privileges being assigned, and issuers must know the security policies being configured. Technically, the credentials stored at subjects will enable the local backward tracking in delegation network.

Lemma 4. Let e be the edge $A.r \xleftarrow{\square} B.r_b$ and $e \in G_A|_\tau$, if $B.r_b \twoheadleftarrow C.r_c \in G_B|_\tau$ then $B.r \twoheadleftarrow C.r_c \in G_A|_\tau$, where no CDA instances are running on A and B at τ.

Lemma 5. Let v be an intersection contains $B.r_b.\square$ and there is an edge from v to $A.r$ in $\mathrm{G}_A|\tau$, if $B.r_b \twoheadleftarrow C.r_c \in \mathrm{G}_B|\tau$ then $B.r_b \twoheadleftarrow C.r_c \in \mathrm{G}_A|\tau$, where no CDA instances are running on A and B at τ.

Given entities A and B, we use $\Sigma_{A:B}$ to denote the union of Σ_A and Δ_B, and $\mathrm{G}_{A:B}$ to denote the credential graph of $\Sigma_{A:B}$. Lemma 4 and 5 can be used to prove the soundness and completeness of CDA, with respect to the semantics of ScoRT.

Theorem 2 (Soundness and Completeness of CDA). Given an entity D and a role $A.r$, then $\Sigma_{A:D}|\tau \hookrightarrow \mathrm{mem}(D, A.r)$ if and only if $\Sigma|\tau \hookrightarrow \mathrm{mem}(D, A.r)$, where Σ is the set of all credentials and no CDA instances are running on A and D at τ.

Given entities A and B, if B requests the resources controlled by A. By theorem 2, A need not search any delegation credentials to make sound and complete authorization decisions. But the theorem assumes that Δ_B (all type-1 credentials issued to B) should be available to A, and retrieval of Δ_B is beyond CDA.

3.2 Credential Revocation

Two kinds of revocation can be provided in ScoRT: revocation on expiration and revocation on demands. Expiration time is the most efficient method for credential revocation, especially for short-term credentials. ScoRT can use revocation messages to enable revocation on demands. Given a credential c of the form $A.r \leftarrow b$, its revocation message can be denoted as $A.r \nleftarrow b$, which is also signed by A. The entity who receives the message $A.r \nleftarrow b$ will delete the credential $A.r \leftarrow b$ from its local credential repository. Revocation of the credential c involves the following four operations:

1. ScoRT stores each authorization credential at its issuer and subject, so if b is an entity, A sends the message $A.r \nleftarrow b$ to A and b.
2. ScoRT stores each delegation credential at its issuer and subject, so if b is not an entity, A sends the message $A.r \nleftarrow b$ to A and the subject entities in b.
3. ScoRT stores the affiliation graphs of each role at its defining entity, so if b is not an entity, A sends the message $A.r \nleftarrow b$ to entities that uses $A.r$ to define credentials.
4. The affiliation graph should be deleted from the local credential repositories if its root is isolated from all roles defined by the local entity.

4 Complexity Estimation

Affiliation graphs will be transferred among entities in credential distribution processes. We have shown that the worst space complexity of affiliation graphs is liner to the number of delegation credentials. This section gives more practical analysis on complexities of affiliation graphs and CRA algorithm based on the branching matrix model of layered delegation networks [1], which can be formally defined as follows:

$$\mathrm{DN} = (F, B, R)$$

where F is an n-dimensional forward branching matrix, B is an n-dimensional backward branching matrix, R is an n-dimensional vector and $R^{(i)} \cdot F^{(j,i)} = R^{(j)} \cdot B^{(i,j)}$. Let G_{DN} be a credential graph which complies with DN:

- $R^{(i)}$ is the number of entity nodes and role nodes at layer i in G_F ;
- $F_{n-1}^{(i,j)}$ is the average of trust edges from a node in layer i to layer j, and $F^{(i,n)}$ is the average of authorization edges from a node in layer i to layer n;
- $B_{n-1}^{(i,j)}$ is the average of trust edges to a node in layer i from layer j, $B^{(n,j)}$ is the average of authorization edges to a node in layer n from layer j;

where $F_{n\text{-}1}$ is a matrix with n-1 dimensions and $F_{n-1}^{(i,j)} = F^{(i,j)}$, $i, j\in[1, n\text{-}1]$. Let $A.r$ be the role node at layer 1 in G_{DN} , we use $\mathrm{G}_{\mathrm{DN}}^{A.r}$ to denote the credential graph for $A.r$ and sizeof($\mathrm{G}_{\mathrm{DN}}^{A.r}$) to denote the number of credentials in $\mathrm{G}_{\mathrm{DN}}^{A.r}$. By definition 3, sizeof($\mathrm{G}_{\mathrm{DN}}^{A.r}$) = $|\,\mathrm{E}_{\mathrm{DN}}^{A.r}.\mathrm{TE}\,|$ - $|\,\mathrm{E}_{\mathrm{DN}}^{A.r}.\mathrm{IE}\,|$, where $\mathrm{G}_{\mathrm{DN}}^{A.r}$ is the edge set of $\mathrm{G}_{\mathrm{DN}}^{A.r}$. Apparently sizeof($\mathrm{G}_{\mathrm{DN}}^{A.r}$) is bounded by $|\,\mathrm{E}_{\mathrm{DN}}^{A.r}.\mathrm{TE}\,|$ which can be computed by the following equation:

$$|\,\mathrm{E}_{\mathrm{DN}}^{A.r}.\mathrm{TE}\,| = V_n \cdot (I_n + \Sigma_{k=1}^{n-1} F^k)\cdot U_n^T \tag{1}$$

where V_n is an n-dimensional unit row vector, I_n is an n-dimensional identity matrix, U_n is an n-dimensional row vector and $U_n^{(1)}$ =1 and $U_n^{(i)}$ =0 for $i\in[2, n]$. The equation shows that the worst complexity of affiliation graphs may increase exponentially on the scale of delegation networks. However, delegation networks in practical systems usually have specific structure models [1] with acceptable costs.

Now we analyze the space and communication costs based on two sample delegation networks derived from existing researches [1, 16]. Compared with the delegation structures in the EHR system [3], these sample networks seem quite complex. Given a delegation network DN and a role $A.r$ at layer one in G_{DN} , then $\mathrm{G}_{\mathrm{DN}}^{A.r}$ only contains the nodes at first n-1 layers. According to equation (1), we can compute the trust edges in $\mathrm{G}_{\mathrm{DN}}^{A.r}$ which can be reached within specified delegation steps: AGN($F_{n\text{-}1}$, d) is the number of trust edges in $\mathrm{G}_{\mathrm{DN}}^{A.r}$ which can be reached by a legal path from $A.r$ within d steps. Similarly, we use CGN(F, d) to denote the number of both trust and authorization edges in G_{DN} which can be reached by a legal path from $A.r$ within d steps.

$$\mathrm{AGN}(F_x, d) = V_x \times (I_x + \Sigma_{k=1}^{d} F_x^k\,) \times U_x^T \text{ - } 1 \tag{2}$$

$$\mathrm{CGN}(F, d) = V_n \times (I_n + \Sigma_{k=1}^{d} F^k\,) \times U_n^T \tag{3}$$

where $x\in[0, n\text{-}1]$, $d\in[0, n\text{-}1]$, and because credentials issued to layer 5 are authorization credentials, we have AGN($F_{n\text{-}1}$, n-1) = AGN($F_{n\text{-}2}$, n-2). Usually, practical applications only permit delegation paths with small depths, such as 2 and 3, and the depths of more than 4 are rarely considered [15]. Let u be the upper bound of depths, the length of legal paths in affiliation graphs is u-1 at most. By line 17 in CDA, each affiliation graph being transferred contains at most AGN($F_{n\text{-}1}$, u-2) trust edges. Given a root CDA instance α, by lines 14~16 of CDA, α transfers the affiliation graphs over the edges covered by the backward searching processes in [1], which is BGN($B_{n\text{-}1}$, u-1). Therefore the worst communication cost between $[\alpha^{-}, \alpha^{+}]$ is CDN(F, B):

$$\mathrm{CDN}(F, B) = \mathrm{AGN}(F_{n\text{-}1}, u\text{-}2) \times \mathrm{BGN}(B_{n\text{-}1}, u\text{-}1) \tag{4}$$

$$\mathrm{BGN}(B_x, d) = V_d \times (I_d + \Sigma_{k=1}^{d} B_x^k\,) \times {Z_d}^T \text{ - } 1 \tag{5}$$

where Z_d is an d-dimensional row vector, $Z_d^{(d)}=1$ and $Z_d^{(i)}=0$ for $i\in[1, d-1]$. Now we estimate the complexities of affiliation graphs and CDA communication costs based on two sample delegation networks where the upper bound of delegation depth is 4.

CASE1: Consider the sample delegation network defined in [1], as shown in the first three columns of Table 1. The nodes at layer 1 ~ 4 are roles, while the nodes at layer 5 are entities. The credentials from layer i to layer j ($1 \le i \le j \le 4$) are delegation credentials. The credentials issued to the entities at layer 5 are authorization credentials.

Table 1. Cost Estimation based on the Delegation Network in [1]

Amt. of credentials per role (F)		From layer					Amt. of roles(R)	AGN, CGN, %	CDN(F, B)
		1	2	3	4	5			
To layer	1	0	0	0	0	0	10	0, 0, -	1722 (21×82)
	2	1	2	0	0	0	5	4, 14, 29%	
	3	1	2	2	0	0	2	21, 86, 24%	
	4	2	2	5	2	0	20	83, 418, 20%	
	5	10	5	10	20	0	2000	83, 1678, 4.9%	

By definition of delegation network, the backward branching matrix B is [0, 2, 5, 1, 0.05; 0, 2, 5, 0.5, 0.0125; 0, 0, 2, 0.5, 0.01; 0, 0, 0, 2, 0.2; 0, 0, 0, 0, 0]. Therefore CDN(F, B) = AGN(F_4, 2)×BGN(B_4, 3)=1722. According to the 4th column in table 1, AGN(F_i, i) is relatively small compared with CGN(F, i), $i\in[0, 4]$. The largest affiliation graph in credential repositories is AGN(F_3, 3) with only 83 credentials. The largest affiliation graph to be transferred on network contains only 21 credentials and the upper bound of total credentials to be transferred in one root CDA instance is 1722.

CASE2: Consider the credentials in Example 1. The estimated delegation network is given in Table 2. The layer model is: EPub.discount is at layer 1; EOrg.preferred and ACM.member are at layer 2; StateU.student, RegB.student and Alice are at layer 3, 4, 5 respectively. Similar to the computation process in case 1, CDN(F, B) = 2772. But in case 2, AGN(F_i, i) is a much small portion of CGN(F, i), $i\in[0, 4]$. This is because that the number of authorization credentials in case 2 is practically increased. The worst complexity of affiliation graphs does not exceed 466.

Table 2. Cost Estimation based on the Delegation Network for Example 1

Amt. of credentials per role (F)		From layer					Amt. of roles (R)	AGN / CGN / %	CDN(F, B)
		1	2	3	4	5			
To layer	1	0	0	0	0	0	10	0, 0, -	2772 (231×12)
	2	5	0	0	0	0	5	46, 146, 3.2%	
	3	20	10	0	0	0	50	231, 22831, 1.0%	
	4	20	15	2	1	0	500	466, 163066, 0.29%	
	5	100	100	100	1000	0	20000	466, 398301, 0.12%	

In practical systems, a credential mainly contains two keys (1024 bits each) and at most three role names (256 bits each). Together with accessories such as time and tags, the size of one credential can be around 1K bytes. Therefore, the worst communication costs in above two cases are about 1,722K bytes and 2,772K bytes respectively.

5 Related Work

Delegation-based authorization complicates the distribution and retrieval of credentials in decentralized TM systems. This section gives more detailed analysis and comparison of related works in this area.

RT [16, 17] is an influential role-based TM system and deeply studies the credential discovery problem. Li et al designs a graph-based distributed discovery algorithm and a type system to define storage policies. However, the credentials storage policies are difficult to configure which can only be tackled by experts, and the linked roles in RT make the algorithm log-space P-complete. The graph model in [16] does not define edges between intersections and roles, and can not be used to define affiliation graphs in ScoRT. The derived intersection edges in [16] are denoted by the weights of intersections edges in our graph model, which provides more intuitionistic graphical interfaces for security administrators. Furthermore, ScoRT provides delegation control with depth 1, 2 and ∞, where the depth 2 is enabled by the trust scope tag ■, which covers a wide range of practical delegation scenarios.

Aura firstly studies the graph-based methods for credential searching in SPKI certificate databases [1] and suggests that backward searching usually perform much faster than forward searching. Aura proposes the branch matrices model for practical delegation structures, which are used in this paper to evaluate the complexity of our approach. Based on Aura' work, a DNS-based storage scheme for SPKI certificates and certificate retrieval algorithms are proposed [13]. They distinguish four kinds of SPKI certificates and the certificates are stored by issuer sites or subject sites according to their types.

QCM [12] was the first TM system to consider automated credential retrieval: if QCM engine is not given the required credential, the system will retrieve it from the specified server. It provides a language-based framework for automatic retrieval of certificates and distribution of revocation information. SD3 [14] is a successor of QCM based on distributed Datalog which supports certified evaluation and recursive policies. Cassandra [2] borrows the credential retrieval mechanism in SD3 and provides various trust negotiation and credential retrieval strategies.

Table 3 shows an overall comparison with existing approaches. Here clients are resource requesters, and servers are authorizers that provide resources. We distinguish four kinds of credential retrieval models. The client-push and server-pull models are similar to the traditional push and pull models in PMI [11]. ScoRT introduces the server-push model in which servers push credentials to other servers according to credential storage policies. Some TM systems support client-pull model where clients retrieve credential chains from network before push them to servers.

Table 3. Credential Retrieval Models for Distributed Authorization

Retrieval Models	SPKI-DNS	QCM	RT	Cassandra	ScoRT
client-pull	Yes/AC		Yes/AC		
client-push	Yes/AC		Yes/AC		Yes/AC
server-pull	Yes/AC	Yes/AC	Yes/AC	Yes/AC	Yes/SA
server-push				Yes/AC	Yes/SA

We argue that the authorization tasks can be divided into two stages: access control (AC) stages and security administration (SA) stages. AC stages are usually performance sensitive because AC decisions are used to reply login requests. While at SA stages, the reasonable delay caused by security configuration are commonly accepted. ScoRT moves the credential retrieval tasks from AC stages to SA stages and helps to lift the overall system performance. Furthermore, ScoRT can be extended to enforce various credential privacy policies in CDA at SA stages.

6 Conclusion

This paper initiates the research on decentralized authorization that eliminates the process of searching credentials from networks when making access control decisions. The main contributions of our work include: (1) a scoped-role based TM system named ScoRT with more refined delegation control is proposed, which can be regarded as a generalized extension of both role-based and capability-based TM systems; (2) a novel weighed graph-based credential affiliation model is proposed to guide the distributed storage, retrieval and revocation of credentials, which can provide more friendly graphical UIs; (3) a distributed credential management framework is proposed to enable decentralized authorization without searching credentials from network when making access decisions, and thus greatly increases the system performance; (4) first attempt to use Aura's delegation network model to analyze the space complexities of distributed credential management processes, and the preliminary analysis results show that the costs of our approach are mainly acceptable. Further research on ScoRT are necessary, such as fault tolerance, retrieval of authorization credentials, privacy in credential retrieval, and more effective revocation mechanism that can work consistently with CDA.

Acknowledgement

This research is supported by the National 863 Grand project 2007AA010301 and the National 973 Basic Research program 2005CB321800.

References

1. Aura, T.: Fast access control decisions from delegation certificate databases. In: Boyd, C., Dawson, E. (eds.) ACISP 1998. LNCS, vol. 1438, pp. 284–295. Springer, Heidelberg (1998)

2. Becker, M.Y., Sewell, P.: Cassandra: Flexible Trust Management, Applied to Electronic Health Records. In: Proceedings of the 17th IEEE Computer Security Foundations Workshop (2004)
3. Becker, M.Y.: A formal security policy for an NHS electronic health record service. UCAM-CL-TR 628, University of Cambridge, Computer Laboratory, p. 81 (March 2005)
4. Becker, M.Y., Fournet, C., Gordon, A.D.: Design and Semantics of a Decentralized Authorization Language. In: 20th IEEE Computer Security Foundations Symposium, pp. 3–15 (2007)
5. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 164–173. IEEE Computer Society Press, Los Alamitos (1996)
6. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance-checking in the PolicyMaker trust management system. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 254–274. Springer, Heidelberg (1998)
7. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The KeyNote trust-management system, version 2. IETF RFC 2704 (September 1999)
8. Clarke, D., Elien, J.E., Ellison, C., Fredette, M., Morcos, A., Rivest, R.L.: Certificate chain discovery in SPKI/SDSI. Journal of Computer Security 9(4), 285–322 (2001)
9. Elley, Y., Anderson, A., Hanna, S., Mullan, S., Perlman, R., Proctor, S.: Building certification paths: Forward vs. reverse. In: Proceedings of the 2001 Network and Distributed System Security Symposium (NDSS 2001), pp. 153–160. Internet Society (February 2001)
10. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI certificate theory. IETF RFC 2693 (September 1999)
11. Farrell, S., Housley, R.: An Internet Attribute Certificate Profile for Authorization, RFC3281 (April 2002)
12. Gunter, C., Jim, T.: Policy-directed certificate retrieval. Software: Practice & Experience 30(15), 1609–1640 (2000)
13. Hasu, T., Kortesniemi, Y.: Implementing an SPKI Certificate Repository within the DNS. In: International Workshop on Public-Key Cryptography, PKC (2000)
14. Jim, T.: SD3: A trust management system with certified evaluation. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, pp. 106–115. IEEE Computer Society Press, Los Alamitos (2001)
15. Li, N.: Delegation Logic: A Logic-based Approach to Distributed Authorization. PhD thesis, New York University, New York (2000)
16. Li, N., Winsborough, W.H., Mitchell, J.C.: Distributed credential chain discovery in trust management (extended abstract). In: Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8), pp. 156–165. ACM Press, New York (2001)
17. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society Press, Los Alamitos (2002)
18. Mao, Z., Li, N., Winsborough, W.H.: Distributed Credential Chain Discovery in Trust Management with Parameterized Roles and Constraints. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 159–173. Springer, Heidelberg (2006)
19. Nilsson, U., Małuszyński, J.: Logic, Programming and Prolog, 2nd edn. John Wiley & Sons Ltd., Chichester (1995)

Appendix A

A. 1 Proof of Lemma 1

Proof. We first prove the if part. Do induction on the length k of $A.r \twoheadleftarrow D$. If k is one, then $A.r \longleftarrow D \in G_\Sigma$ and $A.r \leftarrow D \in \Sigma$. By ScoRT semantics, $\Sigma \hookrightarrow \mathrm{mem}(D, A.r)$ is true. If k is greater than one, suppose n is the node in $A.r \twoheadleftarrow D$, then there is an edge $A.r \xleftarrow{w} n$ in $A.r \twoheadleftarrow D$ and the length of $n \twoheadleftarrow D$ is k-1. By definition of credential graph, n is either a role or a intersection. If n is a role, by induction assumption, $\Sigma \hookrightarrow \mathrm{mem}(D, n)$ and thus $P_\Sigma \vdash \mathrm{m}(D, n, \square)$. Because there is an edge $A.r \xleftarrow{w} n$ in G_Σ, the credential $A.r \leftarrow n.s$ must be in Σ and thus P_Σ contains the rule $\mathrm{m}(x, A.r, \square) \leftarrow \mathrm{m}(x, n, w)$. If w is $\square$ then $P_\Sigma \vdash \mathrm{m}(D, A.r, \square)$ and $\Sigma \hookrightarrow \mathrm{mem}(D, A.r)$; otherwise if w is $\blacksquare$, because $A.r \twoheadleftarrow D$ is a path in G_Σ, then k must equal to 2, therefore $\Sigma \hookrightarrow \mathrm{mem}(D, A.r)$ follows too. If n is a intersection $n_1 \wedge n_2$, then G_Σ contains the edge $A.r \xleftarrow{w} n$ and the paths $n_1 \twoheadleftarrow D$ and $n_2 \twoheadleftarrow D$ where $D \in w$, and thus P_Σ contains the rule $\mathrm{m}(x, A.r, \square) \leftarrow \mathrm{m}(x, n_1, s_1), \mathrm{m}(x, n_2, s_2)$. By the induction assumption, $\Sigma \hookrightarrow \mathrm{mem}(D, n_1)$ and $\Sigma \hookrightarrow \mathrm{mem}(D, n_2)$. Similar to the case when n is a role, $\Sigma \hookrightarrow \mathrm{mem}(D, A.r)$ can be proved.

Now we consider the only if part. By $\Sigma \hookrightarrow \mathrm{mem}(D, A.r)$, $P_\Sigma \vdash \mathrm{m}(D, A.r, \square)$ follows. There is a sequence of proof steps for $\mathrm{m}(D, A.r, \square)$. Do induction on the length s of the proof steps. If s equals one, then the rule R1 is used in the proof and the edge $A.r \longleftarrow D$ is in G_Σ. $A.r \twoheadleftarrow D \in G_\Sigma$ follows. If s is greater than one, suppose α is the rule used at the last proof step which is generated by T2 or T3. If α is a T2 rule, i.e. α is $\mathrm{m}(x, A.r, \square) \leftarrow \mathrm{m}(x, B.r_1, s)$, then $P_\Sigma \vdash \mathrm{m}(x, B.r_1, s)$ and by induction assumption $B.r_1 \twoheadleftarrow D \in G_\Sigma$. If s is $\square$, then $A.r \xleftarrow{\square} B.r_1$ is in G_Σ and $A.r \twoheadleftarrow D \in G_\Sigma$. Otherwise, if s is $\blacksquare$ then $A.r \xleftarrow{\blacksquare} B.r_1$ is in G_Σ. Because $P_\Sigma \vdash \mathrm{m}(D, A.r, \square)$, then there must be a rule $\mathrm{m}(D, B.r_1, \blacksquare)$ in P_Σ and the edge $B.r_1 \xleftarrow{\blacksquare} D$ is in G_Σ. Therefore $A.r \twoheadleftarrow D \in G_\Sigma$. If α is a T3 rule, i.e. α is $\mathrm{m}(x, A.r, \square) \leftarrow \mathrm{m}(x, B_1.r_1, s_1), \mathrm{m}(x, B_2.r_2, s_2)$, then $P_\Sigma \vdash \mathrm{m}(x, B_1.r_1, s_1)$ and $P_\Sigma \vdash \mathrm{m}(x, B_2.r_2, s_2)$. By induction assumption, $B_1.r_1 \twoheadleftarrow D \in G_\Sigma$ and $B_2.r_2 \twoheadleftarrow D \in G_\Sigma$ are true. Similar to the case of T2 rule, $A.r \twoheadleftarrow D \in G_\Sigma$ follows.

A. 2 Proof of Lemma 2

Proof. Let ξ be the legal path $A.r \twoheadleftarrow n$ in G_Σ. Do induction on the length s of ξ. When s is one, ξ is an edge. By line 3 and 4 of AffG, ξ will be added into $G_\Sigma^{A.r}$. When s is greater than one, then there is a node n' in ξ. Let e be the edge $n' \xleftarrow{w} n$ in ξ. Consider the path $A.r \twoheadleftarrow n'$ which is a sub-path of ξ. By induction assupmtion, all edges in $A.r \twoheadleftarrow n'$ are in $G_\Sigma^{A.r}$. Now we prove e is also in $G_\Sigma^{A.r}$. Let e' be $n'' \xleftarrow{w'} n'$ which is the last edge in $A.r \twoheadleftarrow n'$. Because $\xi \in G_\Sigma$, w' is either $\square$ or $\varnothing$. By definition of AffG, e' can only be added by line 4 or line 7. In the first case, if w' is $\square$, then AffG(n', ns, es) will add e into $G_\Sigma^{A.r}$; if w' is $\varnothing$, then e will also be added by line 7. In the second case, w' must be $\square$ and AffG(n', ns, es) will add e into $G_\Sigma^{A.r}$.

A. 3 Proof of Lemma 3

Proof. The if part is obvious because $G_{\Sigma}^{A.r\,:\,D}$ is a sub-graph of G_{Σ}. We now prove the only if part. For each edge e in $A.r \twoheadleftarrow D$ where e is $n \xleftarrow{w} n'$, if n is a role, by definition of legal paths, $A.r \twoheadleftarrow n \in G_{\Sigma}$. By lemma 2, e is in $G_{\Sigma}^{A.r}$. If n is a intersection $n_1 \wedge n_2$, there must be a role n'' that $n'' \xleftarrow{w'} n$ is in $A.r \twoheadleftarrow D$ and $A.r \twoheadleftarrow n'' \in G_{\Sigma}$. By definition of legal paths, $A.r \twoheadleftarrow n_1 \in G_{\Sigma}$ and $A.r \twoheadleftarrow n_1 \in G_{\Sigma}$ are true. By lemma 2, the edges $n'' \xleftarrow{w'} n$, $n \xleftarrow{s_1} n_1$ and $n \xleftarrow{s_2} n_2$ are in $G_{\Sigma}^{A.r}$, and e is in $G_{\Sigma}^{A.r}$ because e is either $n \xleftarrow{s_1} n_1$ or $n \xleftarrow{s_2} n_2$. Therefore $A.r \twoheadleftarrow D \in G_{\Sigma}^{A.r\,:\,D}$ is true.

A. 4 Proof of Lemma 4

Proof. Let ξ be the path $B.r_b \twoheadleftarrow C.r_c$ and $n_1, \ldots, n_s$ are nodes appearing in ξ along the reverse direction of the path where n_1 is $B.r_b$, n_s is $C.r_c$ and s is greater than one. Consider two CDA instances α and β: α is a root instance where $e \notin G_A|_{\alpha^-}$ and $e \in G_A|_{t}$, $\alpha^+ \le t \le \tau$; β is a root instance where $\xi \notin G_B|_{\beta^-}$ and $\xi \in G_B|_{t}$, $\beta^+ \le t \le \tau$. By definition of β, there exits a path $n_1 \twoheadleftarrow n_i$ in $G_B|_{\beta^-}$ and there is no edge from n_{i+1} to n_i, $0<i<s$. Therefore β stores $n_i \xleftarrow{w} n_{i+1}$ into $G_B|_{\beta^+}$ and n_i is a role[1]. If α is the same instance as β, α will retrieve $n_i \xleftarrow{w} n_{i+1}$ from B by line 6. which is in contradiction to $n_i \xleftarrow{w} n_{i+1} \in G_B|_{\beta^+}$, therefore α is not β. Now we need to prove that $B.r \twoheadleftarrow C.r_c \in G_A|_{\tau}$ is true in both cases: α runs before β and α runs after β.

Let c be the credential $A.r \leftarrow B.r_b.\square$. Do induction on s and consider the base case when s is 2, then ξ is a trust edge and let c' be the credential $B.r_b \leftarrow C.r_c.s_c$. In the first case, α stores c at A and send it to B by line 8. And then β stores c' at B and uses line 16 to send c' to A, and thus Edges(c') is at A at time τ. In the second case, β firstly stores Edges(c') at B, then α stores Edges(c) at A and use line 7 to pull $G_B^{B.r_b}$ from B. By lemma 2, Edges(c')$\in G_B^{B.r_b}$. Therefore, $B.r \twoheadleftarrow C.r_c \in G_A|_{\tau}$ in both cases.

Consider the induction step where s is greater than 2. In the first case, $\alpha^+ < \beta^- \le \tau$, α stores c at A and send it to B by line 8, i.e., $e \in G_A|_{\alpha^+}$. By $n_1 \twoheadleftarrow n_i \in G_B|_{\beta^-}$ and $e \in G_A|_{\beta^-}$, $B.r \twoheadleftarrow n_i \in G_A|_{\beta^-}$ is true by induction assumption. Because β stores $n_i \twoheadleftarrow n_s$ into $G_B|_{\beta^-}$, there must be a instance β' running on B that uses line 1 to add $n_i \twoheadleftarrow n_s$ into $G_B|_{\beta^+}$. Because $e \in G_A|_{\alpha^+}$, β' will use line 16 or 17 to send $G_B^{B.r_b}$ to A. By lemma 2, $n_i \twoheadleftarrow n_s \in G_B^{B.r_b}$. In the second case, $\beta^+ < \alpha^- \le \tau$, β firstly stores $n_1 \twoheadleftarrow n_s$ at B, then α uses line 7 to pull $G_B^{B.r_b}$ from B. By lemma 2, $n_1 \twoheadleftarrow n_s \in G_A^{B.r_b}|_{\alpha^+}$. Therefore $B.r \twoheadleftarrow C.r_c \in G_A|_{\tau}$ is true in both cases.

A. 5 Proof of Lemma 5

Proof. Let e be the intersection edge from v to $A.r$. Let ξ be the path $B.r_b \twoheadleftarrow C.r_c$ and $n_1, \ldots, n_s$ are nodes appearing in ξ along the reverse direction of the path where n_1 is $B.r_b$, n_s

[1] Otherwise n_i is an intersection $e_{i1}.s_{i1} \wedge e_{i2}.s_{i2}$, and n_{i+1} is either e_{i1} or e_{i2}; by lines 1 and 2 of CDA, there are two edges from e_{i1} and e_{i2} to n_i in $G_B|_{\beta^-}$ which is paradoxical because there is no edges from n_{i+1} to n_i.

is $C.r_c$ and s is greater than one. Consider two CDA instances α and β: α is a root instance where $e \notin \mathrm{G}_A|_{\alpha^-}$ and $e \in \mathrm{G}_A|_{\mathrm{t}}$, $\alpha^+ \leq \mathrm{t} \leq \tau$; β is a root instance where $\xi \notin \mathrm{G}_B|_{\beta^-}$ and $\xi \in \mathrm{G}_B|_{\mathrm{t}}$, $\beta^+ \leq \mathrm{t} \leq \tau$. By definition of β, there exits a path $n_1 \twoheadleftarrow n_i$ in $\mathrm{G}_B|_{\beta^-}$ and there is no edge from n_{i+1} to n_i, $0<i<s$. Therefore $n_i \xleftarrow{w} n_{i+1} \in \mathrm{G}_B|_{\beta^+}$. By similar analysis in lemma 4, n_i must be a role and α is the different instance from β. Now we need to prove that $B.r \twoheadleftarrow C.r_c \in \mathrm{G}_A|_\tau$ is true in both cases: α runs before β and α runs after β.

Let c be the credential for e. Do induction on s and consider the base case when s is 2, then ξ is a trust edge and let c' be the credential $B.r_b \longleftarrow C.r_c.s_c$. In the first case, α stores c at A and send it to B by line 12. And then β stores Edges(c) at B and uses line 16 to send c to A, and thus Edges(c') is at A at time τ. In the second case, β firstly stores Edges(c') at B, then α stores c' at A and use line 11 to pull $\mathrm{G}_B^{B.r_b}$ from B. By lemma 2, Edges(c')$\in \mathrm{G}_B^{B.r_b}$. Therefore, $B.r \twoheadleftarrow C.r_c \in \mathrm{G}_A|_\tau$ in both cases.

Consider the induction step where s is greater than 2. In the first case, α stores Edges(c) at A and send c to B by line 12. By $n_1 \twoheadleftarrow n_i \in \mathrm{G}_B|_{\beta^-}$ and $e \in \mathrm{G}_A|_{\beta^-}$, $B.r \twoheadleftarrow n_i \in \mathrm{G}_A|_{\beta^-}$ is true by induction assumption. Because β stores $n_i \twoheadleftarrow n_s$ into $\mathrm{G}_B|_{\beta^-}$, there must be a instance β' running on B that uses line 1 to add $n_i \twoheadleftarrow n_s$ into $\mathrm{G}_B|_{\beta^+}$. Because $e \in \mathrm{G}_A|_{\alpha^+}$, β' will use line 16 or 17 to send $\mathrm{G}_B^{B.r_b}$ to A. By lemma 2, $n_i \twoheadleftarrow n_s \in \mathrm{G}_B^{B.r_b}$. In the second case, $\beta^+ < \alpha^- \leq \tau$, β firstly stores $n_1 \twoheadleftarrow n_s$ at B, then α uses line 7 to pull $\mathrm{G}_B^{B.r_b}$ from B. By lemma 2, $n_1 \twoheadleftarrow n_s \in \mathrm{G}_A^{B.r_b}|_{\alpha^+}$. Therefore $B.r \twoheadleftarrow C.r_c \in \mathrm{G}_A|_\tau$ is true in both cases.

A. 6 Proof of Theorem 2

Proof. $\Sigma|_\tau \hookrightarrow \mathrm{mem}(D, A.r)$ follows from $\Sigma_A|_\tau \hookrightarrow \mathrm{mem}(D, A.r)$ since $\Sigma_A \subseteq \Sigma$. We only need to prove the if part. By lemma 1 we need to prove $A.r \twoheadleftarrow D \in \mathrm{G}_{A:D}|_\tau$ if $A.r \twoheadleftarrow D \in \mathrm{G}_\Sigma|_\tau$.

Do induction on the length k of $A.r \twoheadleftarrow D$. If k equals one, then $A.r \twoheadleftarrow D$ is $A.r \longleftarrow D$. $A.r \twoheadleftarrow D \in \mathrm{G}_{A:D}|_\tau$ follows. If k equals two, $A.r \twoheadleftarrow D$ can be denoted as $A.r \xleftarrow{w} B.r_b \longleftarrow D$. By $A.r \twoheadleftarrow D \in \mathrm{G}_\Sigma|_\tau$, there must be a root CDA instance that adds $A.r \xleftarrow{w} B.r_b$ into G_A before τ. Therefore $A.r \twoheadleftarrow D \in \mathrm{G}_{A:D}|_\tau$ is true.

Consider the induction step when k is greater than two. Suppose n and n' are nodes in $A.r \twoheadleftarrow D$, where $A.r \xleftarrow{w} n$ and $n' \longleftarrow D$ are two edges in $\mathrm{G}_A|_\tau$, then the length of $n \twoheadleftarrow D$ is k-1. By definition of legal paths, w must be □. There must be a root CDA instance that uses line 2 to add Edges($A.r \longleftarrow n$) into $\mathrm{G}_A|_\tau$ and thus $A.r \xleftarrow{w} n$ is in $\mathrm{G}_A|_\tau$. By definition of credential graph, n can be a role or an intersection. If n is a role $B.r_b$, by induction assumption we have $B.r_b \twoheadleftarrow D \in \mathrm{G}_{B:D}|_\tau$. By lemma 3, $B.r \twoheadleftarrow n' \in \mathrm{G}_A|_\tau$ is true and therefore $A.r \twoheadleftarrow D \in \mathrm{G}_{A:D}|_\tau$. If n is an intersection $B.r_b \wedge B'.r_{b'}$, then there are two edges $n \xleftarrow{w_1} B.r_b$ and $n \xleftarrow{w_2} B'.r_{b'}$ in $\mathrm{G}_\Sigma|_\tau$. Consider paths $B.r_b \twoheadleftarrow n'$ and $B'.r_{b'} \twoheadleftarrow n'$ in $\mathrm{G}_\Sigma|_\tau$, by induction assumption, $B.r_b \twoheadleftarrow D \in \mathrm{G}_{B:D}|_\tau$ and $B'.r_{b'} \twoheadleftarrow D \in \mathrm{G}_{B':D}|_\tau$. By lemma 4, we can easily prove that both $B.r_b \twoheadleftarrow D$ and $B'.r_{b'} \twoheadleftarrow D$ are in $\mathrm{G}_{A:D}|_\tau$. Therefore $A.r \twoheadleftarrow D \in \mathrm{G}_{A:D}|_\tau$ follows.

BinHunt: Automatically Finding Semantic Differences in Binary Programs

Debin Gao[1], Michael K. Reiter[2], and Dawn Song[3]

[1] Singapore Management University
dbgao@smu.edu.sg
[2] University of North Carolina at Chapel Hill
reiter@cs.unc.edu
[3] University of California, Berkeley
dawnsong@cs.berkeley.edu

Abstract. We introduce BinHunt, a novel technique for finding semantic differences in binary programs. Semantic differences between two binary files contrast with syntactic differences in that semantic differences correspond to changes in the program functionality. Semantic differences are difficult to find because of the noise from syntactic differences caused by, e.g., different register allocation and basic block re-ordering. BinHunt bases its analysis on the control flow of the programs using a new graph isomorphism technique, symbolic execution, and theorem proving. We implement a system based on BinHunt and demonstrate the application of the system with three case studies in which BinHunt manages to identify the semantic differences between an executable and its patched version, revealing the vulnerability that the patch eliminates.

1 Introduction

Many software vendors make the source code of their programs unavailable. When a program needs to be updated (for patching vulnerabilities and errors), they release a new version in binary format but refuse to disclose details of the changes made. However, it is of great interest for consumers of the software to understand the differences in two versions of the program. Binary difference analysis is one of the most useful techniques in finding these differences.

There are many reasons why consumers want to know the differences between the two binary files. For example, when an update of a program is available, the consumers need to make a decision whether to apply the update. Such a decision may require security analysis of the updated version, which is usually an expensive and time-consuming process. Binary difference analysis can simplify and speed up this process because one can reuse the security properties of the earlier version and focus the analysis on the difference between the two versions. A similar application of binary difference analysis is profile reuse in application development [15]. A large program, with millions of lines of code, may be used in a large variety of ways by different users on different machines. To characterize the program behavior, extensive collection of profile data is required. For example,

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 238–255, 2008.

BMAT is a tool that matches two versions of a binary program to propagate profile information from an older, extensively profiled build to a newer build. Another reason why binary difference analysis is useful is that in many cases the differences in two versions of a program correspond to vulnerabilities in the earlier version that the later version patches. One may be able to find vulnerabilities in a program using binary difference analysis, and subsequently exploit those vulnerabilities and attack consumers who have not applied the update.

Although such binary difference analysis is very useful, it is different from the binary difference tools that we use to produce and apply patches (bsdiff, bspatch, xdelta, jdiff, jpatch, and etc.), because many *syntactic differences* may not correspond to *semantic changes* in the program. In the next section, we will carefully define what syntactic and semantic differences are, but intuitively semantic differences correspond to changes in the program functionality, and are what we seek to identify in this paper.

Finding the semantic differences between two binary files is challenging for many reasons. For example, a small change in the source code may cause the compiler to use a different register allocation in other parts of the program in which the corresponding source code remains the same. Similarly, a small change in the source code may change the size of a small number of basic blocks, which further triggers the compiler to re-order many other basic blocks in the binary file. For these and other reasons, a small change in source code may lead to many changes throughout the binary file. For example, in one of the case studies we report, the patch of the `gzip` program consists of only 5 additional lines of code in one function, but all the 75 non-empty functions in the resulting binary file are changed (see Sect. 6). To find semantic differences between such binaries, we must match semantically identical basic blocks that are nevertheless syntactically different and located differently, and similarly match semantically identical functions. In this paper, we use “match” and “matched pair” to denote a pair of basic blocks (or a pair of functions), one from each binary file, and use “matching” to refer to a set of matched pairs of basic blocks (or functions) in which any basic block (respectively, function) from the first file is matched to at most one from the second file, and vice versa.

We propose a novel technique, called BinHunt, to find the semantic differences between two binary files by analyzing the control flow of the program. The control flow reflects the functionality and seldom changes because of, e.g., different register allocations or basic block re-ordering, making it an attractive feature for finding the semantic differences. BinHunt first constructs a control flow graph (CFG) for each function and a callgraph (CG) for the entire binary file. After that, a customized graph isomorphism algorithm is used to find the best (partial) matchings between functions and between basic blocks. Our graph isomorphism algorithm is more accurate than previous techniques used in binary difference analysis, because its backtracking will replace erroneous matches by better ones, whereas previous approaches are greedy and erroneous matches will propagate through the isomorphism process. The output of this algorithm is a (partial) matching between functions in the two binary files, and a (partial) matching between basic blocks in two matched functions. It also outputs a *matching strength* for each match of functions or basic blocks, which tells how similar the two functions or basic blocks are. The matchings together with the matching strengths tell us where the semantic differences are.

A component of BinHunt is a method to accurately compare two basic blocks using symbolic execution and theorem proving. This novel technique helps determine if two basic blocks are functionally equivalent. It provides a guarantee that if two basic blocks are found to be different by BinHunt, then they must not be functionally equivalent. To the best of our knowledge, this is the first paper introducing such a technique for binary difference analysis. Being able to accurately compare two basic blocks not only improves the accuracy of the graph isomorphism computation, but helps in finding its solution faster.

We have implemented a system based on BinHunt and report on three case studies where we have used it to locate the semantic differences between binaries, which in these cases correspond to vulnerabilities in one version of a binary that was patched in the subsequent version. Note that when used to find software vulnerabilities, BinHunt is not meant to replace existing tools to generate exploits from syntactic binary differences [2], but instead to augment such tools. For example, in one of our case studies, all of the 75 non-empty functions in the binary change syntactically as the result of a very small change to one of the functions in the source code. Rather than applying an exploit-generation tool of Brumley et al. [2] to each of such a large number of syntactic differences to find the one that can be exploited, BinHunt can help identify the one semantic difference so that subsequent analysis can focus on that.

2 Problem Definition and Overview of Our Approach

Given are two binary files, which are usually two versions of the same program, possibly compiled with optimizations for increased performance. We assume that source code of these binary files is not available. We also assume that function names extracted from these binary files are unreliable for the purpose of binary difference analysis, since they can be changed easily.[1]

The outputs of the system are a (partial) matching between functions from the two binary files, a (partial) matching between basic blocks from two matched functions, and a *matching strength* for each pair of matched functions or basic blocks. The matching strength tells how similar the matched functions or basic blocks are. Unmatched functions and unmatched basic blocks, as well as matched functions and matched basic blocks with low matching strengths, constitute the semantic differences found between the two binary files.

The difference between two functions or between two basic blocks could be syntactic and semantic. Syntactic differences refer to differences in the instructions, whereas semantic differences refer to differences in functionality (i.e., input-output behavior). It is possible that a syntactic difference is not semantic.

[1] This is especially important when BinHunt is used to analyze malware, where attackers intentionally make static analysis of the software difficult.

As mentioned in Sect. 1, basic blocks could use different register allocations to perform the same task, which means that the two basic blocks with different instructions may have the same functionality. Similarly, basic block re-ordering may make two functions look very different, but the two functions could provide exactly the same functionality. Here we are concerned with finding semantic differences, and so the output matchings and matching strengths from BinHunt should eliminate functionally identical functions and identical blocks from consideration, by matching them to one another with high matching strengths.

BinHunt uses a graph isomorphism technique, applied to the control flow and call graphs of the two binary files, to find the partial matchings between functions and between basic blocks. In order to compute isomorphisms on graphs, BinHunt first needs to find the control flow of the programs. This is achieved by first disassembling the binary files and locating the code segments. The x86 instructions are then converted into an intermediate representation for constructing the control flow graphs and callgraphs. The graph isomorphism computation in BinHunt uses a novel technique to compare the functionality of basic blocks. This is achieved by symbolically executing the two basic blocks and using a theorem prover to test whether the effects of the basic blocks are always the same. With this novel technique for comparing the basic blocks, BinHunt is able to filter out most syntactic differences that do not correspond to semantic changes in the binary files.

3 System Architecture

Figure 1 shows the overall system architecture. The binary files are first passed to a front-end disassembler, which outputs a sequence of x86 instructions. Next, the x86 instructions are converted into an intermediate representation (IR) for easier and more accurate analysis. The IR is then used for control flow analysis, where the output is a set of control flow graphs (CFGs), one for each function, and a callgraph (CG) for the binary. In the last step, the CFGs and CG of the two binary files are passed to our graph isomorphism engine to find a matching between functions, a matching between basic blocks in matched functions, and matching strengths for each pair of matched functions and basic blocks.

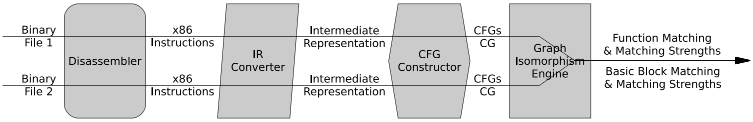


Fig. 1. Overall architecture of BinHunt

3.1 Disassembler

The disassembler parses each binary file and locates the code segments, which are disassembled into a sequence of x86 instructions. We implement a plug-in to

IDA Pro [6], a commercial disassembler, to do this, though other disassemblers could be used as well.

3.2 Intermediate Representation

The x86 computer family has a CISC (Complex Instruction Set Computer) instruction set, and is widely regarded as a very complicated set of instructions. For example, it consists of about 300 instructions; instructions are of variable length; and arithmetic instructions may set status bits, making them have side effects. It has undergone numerous changes over time, most of which were to add new functionality while maintaining backward compatibility. The complex nature of the x86 instruction set makes its static analysis difficult. In fact, some research even takes advantage of this complexity to obfuscate binary executables to improve their resistance to static analysis [10]. Because of this, some researchers choose to perform the analysis on an intermediate representation (IR), instead (e.g., [1]). In this project, we will take a similar approach, converting the x86 instructions into IR first, and then analyzing the IR.

The IR language we use is far simpler. It consists of roughly a dozen different statements, which are type-checked and free of side effects. In some cases, this simplicity does result in a loss of precision, and so our subsequent analysis might conclude that two basic blocks are functionally identical when they are not. However, we have not found examples where this loss of precision hides a semantic change in which we are interested. There are two major benefits of this simplicity. First, it makes the basic block comparison easier and more reliable. It is easier because our symbolic execution and theorem proving are applied on a much simpler set of instructions (see Sect. 4). It is also more reliable because the instruction simplification reduces the language variation in performing the same functionality. For example, two functionally equivalent basic blocks that are syntactically very different in their x86 instructions may look quite similar in their IR, because of the instruction simplification. Second, it makes our control flow analysis much simpler and more scalable (see Sect. 3.3).

3.3 Constructing Control Flow Graphs and Callgraphs

There are many ways of analyzing the IR of two binary files to find their differences. The IR of a program is just a sequence of instructions. Many traditional sequence comparison techniques can be used to find the differences between the two sequences, e.g., dynamic programming [14,12]. However, these techniques are not suitable in this paper, because our objective is not to find all syntactic differences in the instruction level, but to find semantic ones that correspond to changes in the program functionality.

What makes this problem unique is that there could be many changes in the instructions that do not correspond to changes in functionality, as described in Sect. 1. As a result, we propose analyzing the control flow of the programs to find the differences. The control flow of a program is much more resistant to "superficial" changes like different register allocations and basic block re-ordering, and therefore is a more attractive feature for finding semantic differences.

We represent the control flow of a program by a set of control flow graphs and a callgraph. A control flow graph (CFG) consists of a set of nodes each representing a basic block and a set of directed edges representing the control flow among the basic blocks. We construct a CFG for each function found in the binary file. We also construct a callgraph (CG) for each binary file, with the set of nodes corresponding to the functions in the file and the set of directed edges representing calls among the functions.

Note that although we base our binary difference analysis on the control flow of the program, the CFGs and the CG may not contain all necessary information for identifying the differences. For example, two programs may have isomorphic CFGs and CGs, but the nodes in them may be functionally different. Therefore, we still need to compare two corresponding basic blocks to find out if they provide the same functionality. Basic block comparison also helps in graph isomorphism (see Sect. 3.4), and we will detail our novel technique for basic block comparison in Sect. 4.

3.4 Comparing the CGs and the CFGs

A critical part of BinHunt is to compare the CFGs and CGs to find the matchings between corresponding functions and basic blocks. We do this by introducing a new graph isomorphism algorithm based on the backtracking technique.

Suppose we have obtained the CFGs of two functions by disassembling the code segments, converting x86 instructions into IR and analyzing the control flow. The next step is to compare the two CFGs to find nodes that match to each other. This can be conceptualized as the maximum common subgraph isomorphism problem.[2] Intuitively, the maximum common subgraph isomorphism problem is to find the largest common subgraph of two given graphs. Once the maximum common subgraph is found, nodes in the subgraph will correspond to matched basic blocks, and nodes outside of the subgraph will correspond to unmatched ones. Similarly, when given two CGs, we try to find the maximum common subgraph in which the nodes correspond to matched functions, and nodes outside of the subgraph correspond to unmatched functions.

We will detail our algorithm for finding the maximum common subgraph and how to interpret the output of the algorithm in Sect. 5. Note that two matched functions may not be exactly the same. There could be differences within the two functions and therefore we need to find the maximum common subgraph for the two matched functions as well in order to identify the binary differences. Since maximum common subgraph isomorphism is NP-complete, we need to propose an efficient algorithm which works practically for real problems. The efficiency of such an algorithm highly depends on the sequence in which nodes are examined for possible matches. In order to try the most likely matches first, we need to be able to tell whether the basic blocks represented by the nodes are similar. To do

[2] More precisely, we solve the maximum common *induced* subgraph isomorphism problem, which is defined in Sect. 5.1. However, we typically refer to this as simply the "subgraph isomorphism" problem or a similar variant, for simplicity.

this, we propose a novel technique using symbolic execution of the basic blocks and theorem proving. This technique is detailed in Sect. 4.

4 Basic Block Comparison

Basic block comparison is important for two reasons. First, it improves the efficiency of graph isomorphism. As mentioned in Sect. 3.4, the algorithm has better efficiency if the best matches are tried first, which is possible only if there is a way to measure the similarity of two basic blocks accurately. Second, basic block comparison helps identify any semantic differences between matched functions. As noted in Sect. 3.4, two matched functions or basic blocks may not be the same. Accurate basic block comparison can help identify the semantic differences with low false-positive and false-negative rates.

4.1 Symbolic Execution and Theorem Proving

Symbolic execution [7] is a well-known program analysis technique to represent values of program variables with symbolic values instead of concrete (initialized) data and to manipulate expressions involving symbolic values. For each basic block, we first find all the input and output registers and variables. We then use symbolic execution to represent the final values of the output registers and variables. That is, the output values computed by the basic blocks are expressed using the program input symbols. This process is fast since we are dealing with basic blocks, in which instructions are executed sequentially.

To test if two basic blocks are functionally equivalent, we apply a theorem prover to test if the output registers and variables of the basic blocks are the same. The theorem prover we employ is STP [5]. STP is a decision procedure for the satisfiability of quantifier-free formulas in the theory of bit-vectors and arrays that has been optimized for large problems encountered in software analysis applications. We pick the symbolic representation of one register/variable from each basic block and use STP to test if they are equivalent, assuming that the inputs to the basic blocks share the same values.

Symbolic execution with theorem proving helps us determine whether two registers or variables contain the same value after the executions of two basic blocks. However, in general we do not know which registers or variables to pick for testing because the two basic blocks could use different registers or variables to provide the same functionality. We try all pair-wise comparisons and check if there exists a permutation of the registers and variables between the two basic block such that all matched registers and variables contain the same value. If such a permutation exists, we conclude that the two basic blocks are functionally equivalent. With this, our technique ensures that if two basic blocks are found to be different by our technique of symbolic execution and theorem proving, then they must not be functionally equivalent.

Note that this property holds even if the two binary files are compiled using different compilers or compiler options. However, it only holds for basic block comparison and not for function comparison. That is, even if our symbolic execution and theorem proving show that the basic blocks in two functions are different, the two functions may, in fact, be functionally equivalent.

We could perform the same analysis for functions, i.e., we could use symbolic execution to represent outputs of two functions and test if these outputs are equivalent. However, performance becomes an issue as both symbolic execution and theorem proving take a long time to process functions of even moderate size.

4.2 Matching Strength

We define *matching strength* of two basic blocks to denote how similar they are in their functionality. Matching strength is a function of the two basic blocks only, and does not depend on the context in which they execute. Matching strength of two basic blocks that are deemed functionally equivalent (Sect. 4.1) is assigned 1.0 if they use the same registers or 0.9 if they use different ones. If two basic blocks are deemed not functionally equivalent, smaller matching strengths are assigned. The reason why we assign a matching strength of 0.9 for basic blocks using different registers is that our technique evaluates the basic blocks independently. Even if the two basic blocks are functionally equivalent, they may be used in different contexts. This is slightly more likely when the basic blocks use different registers, and therefore we assign a slightly smaller matching strength.

5 Maximum Common Induced Subgraph Isomorphism

Graphs are sets of nodes with edges connecting pairs of nodes. Similarity measurement between graphs can be achieved by graph matching, which is a procedure to identify common subgraphs. The measure of similarity is then given by the size of the maximum common subgraph. This technique has its application to many problems.

Graph matching is known to be a computationally expensive procedure. A number of graph-matching algorithms, both optimal and approximate, have been proposed over the last three decades [11]. Most optimal algorithms for common subgraph isomorphism are based on maximal clique detection in the association graph [9]. However, it is widely accepted that the problem of exact subgraph isomorphism, which is a special case of common subgraph isomorphism with the requirement that the resulting common subgraph coincides with one of the input graphs, is much more efficiently solved by the backtracking algorithm [8,13].

In our problem of binary difference analysis, especially in applications in which the two binary files under analysis are two versions of the same program, the maximum common subgraph is very likely the same or very close to one of the two input graphs. Therefore, we choose an algorithm based on the backtracking technique [8]. Below, we first define the common subgraph problem we are trying to solve. We then describe the basic idea of the backtracking algorithm, and our customizations to it to make it efficient and suitable for binary difference analysis.

5.1 Definitions

Given a graph $G = [V, E]$, graph $H = [W, F]$ is an induced subgraph of G if and only if $W \subseteq V$ and $F = E \cap (W \times W)$. Given two graphs G and H, we define the *maximum common induced subgraph isomorphism* problem as finding the largest induced subgraph of G that is isomorphic to an induced subgraph of H. We call this largest induced subgraph the *maximum common induced subgraph* of G and H. Here "largest" means that the subgraph is largest according to some *subgraph measurement*, which is not necessarily as simple as counting the number of nodes in the subgraph. We define this subgraph measurement in Sect. 5.3.

5.2 Backtracking Algorithm

The backtracking algorithm offers a simple solution to the maximum common induced subgraph isomorphism problem. The algorithm essentially enumerates all possible matches of the nodes from the two input graphs. A property of this algorithm is that an erroneous match added to the result will be replaced by a better match subsequently.

```
Isomorphism $(D, M)$
 1: if Extendable$(D, M)$ then
 2:     $v \leftarrow$ PickAny$(D)$
 3:     $Z \leftarrow$ GetPossibleMatching$(v, D)$
 4:     for all $w \in Z$ do
 5:         $M' \leftarrow M + [v, w]$
 6:         $D' \leftarrow$ Refine$(v, w, D)$
 7:         Isomorphism$(D', M')$
 8:     end for
 9:     $D' \leftarrow$ Refine$(v, \text{null}, D)$
10:     Isomorphism$(D', M)$
11: end if
```

Fig. 2. Isomorphism function

Figure 2 shows the pseudo-code of a recursive version of the backtracking algorithm. If $G = (V, E)$ and $H = (W, F)$ are the input graphs, then D contains all possible pairs of nodes that might still be matched (initially $V \times W$), and M contains matched node pairs (initially empty). We assume that G and H are global variables in Figure 2; all other variables are local and are passed by value. On each entry to `Isomorphism()`, M records the matched node pairs of the partial common induced subgraph found. It first checks whether the solution is extendable. If it is extendable (Line 1), it picks an unmatched node v (Line 2) and assigns Z all possible matches for v (Line 3). For each node w in Z (Line 4), the algorithm extends the solution by adding the match $[v, w]$ to M (Line 5), refining D (Line 6), and recursively calling `Isomorphism()` (Line 7). In the end, the search is complemented by exploring extensions of M that do not include the chosen node v (Line 9 and 10). This last step is necessary because a subgraph not containing v (considered in the last step) may be larger than any subgraphs containing v (considered in earlier steps). With this, `Isomorphism()` completes trying all possible matches of the nodes.

5.3 Customizations to the Backtracking Algorithm

Matching strength and subgraph measurement. In traditional backtracking algorithms [13,8], nodes are labeled (colored) and a possible match is one that consists of nodes with the same label (color). We do not introduce such labels for nodes in our CFGs and CGs, because any nodes in a graph may be matched with any nodes in the other graph in our problem. Instead, we utilize matching strength to guide us which two nodes are more likely to be matched.

The matching strength for two basic blocks was detailed in Sect. 4.2. This is the matching strength used when computing the maximum common induced subgraph of the CFGs of two functions. We define the matching strength of two functions—which is used when computing the maximum common induced subgraph of the CGs of two binaries—to be 1.0 if the instructions (x86 or IR) of the two functions are the same. Otherwise, their matching strength is the *subgraph measurement* divided by the number of nodes in the CFG that has fewer nodes, where *subgraph measurement* is defined as the summation of matching strengths of matched nodes (basic blocks). Subgraph measurement is used not only in defining the matching strength of two functions, but in the definition of maximum common induced subgraph (see Sect. 5.1).

Customizations to improve efficiency. As the backtracking algorithm enumerates all possible matches, it could be very inefficient for large graphs. The functions `Extendable()`, `PickAny()` and `Refine()` in Figure 2 are important in making the algorithm efficient for practical problems.

Function `Extendable()` first makes a prediction on the maximum common induced subgraph assuming all unmatched nodes can be matched perfectly, which is the best possible output given the current matching. If this best possible output is not better than the best subgraph found previously, then further enumeration is not necessary and `Extendable()` simply returns false.

`Extendable()` can improve the efficiency only if good matches are tried first. The earlier the good matches are tried, the more times when `Extendable()` returns false, and therefore the more efficient the algorithm is. So we want to make function `PickAny()` return the best matching candidate first. In BinHunt, `PickAny()` returns the node that has the largest matching strength with nodes in the other graph. `PickAny()` also considers the connectivity (number of predecessors and successors) of the node; i.e., nodes with larger connectivity will be returned if there are multiple nodes with the same maximum matching strength.

`Refine()` updates the set of possible future matches by removing 1) $[v, w]$ (the new match); and 2) other matches that would not conform to the definition of common induced subgraph were they added to M in the future. For example, assume that v has a predecessor v' in G ($(v', v) \in E$), and w' is not a predecessor of w in H ($(w', w) \notin F$). $[v', w']$ has to be removed from D because if

$[v', w']$ is added to M, then the subgraph of G will contain edge (v', v) while the subgraph of H will not contain edge (w', w), making the two subgraphs not isomorphic.

Timeout and output of the backtracking algorithm. With the help of the three functions `Extendable()`, `PickAny()` and `Refine()`, the backtracking algorithm is able to try the best match early and therefore becomes more efficient by quickly terminating the processing of other possible matches that will not result in large subgraphs. However, in some cases where the CFGs are very big, the algorithm may still take too much time to converge.

To have a good balance between efficiency and accuracy, we introduce a timeout on some invocations to `Isomorphism()`. Specifically, when BinHunt tries to find the matching strength for every pair of functions from the two binary files for the input to CG isomorphism, timeouts are enabled. If a timeout is reached, we simply assign a default value to the matching strength. In CG isomorphism, however, the timeout is disabled. Lastly, after the function matching is found, the maximum common induced subgraph is recalculated (with timeout disabled) for matched function pairs that resulted in a timeout in the first step.

The output of BinHunt consists of the (partial) matching between functions from the two binary files, the (partial) matching between basic blocks for matched functions, and the matching strengths for the matched functions and basic blocks. Note that semantic differences correspond to unmatched functions and basic blocks, as well as matched ones with low matching strengths.

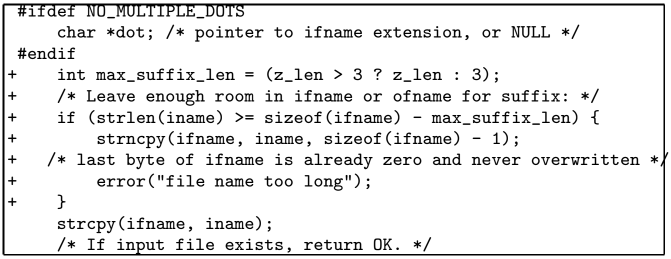
6 Case Studies

We implemented a system for binary difference analysis with the above components and techniques. In this section, we will show the results of using this system in three case studies to discover vulnerabilities in `gzip`, `tar` and `ASP.NET`. We specifically choose these three cases to show how BinHunt performs in complex cases where a small change in one function in the source code leads to substantial syntactic changes in many other functions (the case of `gzip`), when the semantic changes result in change of control flow in the program (the case of `tar`), and when only the binary files are available (the case of `ASP.NET`). For the first two case studies, we obtained the source code of the patched and unpatched versions and compiled them independently to obtain the binary executables for analysis. Once the binary executables were obtained, we made no further use of the source code. For `ASP.NET`, we downloaded the patched and unpatched binaries directly from the software vendor.

Note that in all of our case studies, the only differences between the two versions correspond to patching vulnerabilities. In other cases, the differences may correspond to new features added to the program, which would complicate the discovery of vulnerabilities.

6.1 Buffer Overflow in `gzip`

A release of `gzip` has a vulnerability that causes a buffer overflow with a file name of 1028 bytes or greater. (See http://www.securityfocus.com/bid/3712 for more details.) This overflow could overwrite stack variables and return addresses, possibly resulting in arbitrary code execution. A patch exists for fixing the problem, and is shown in Fig. 3.



```
 #ifdef NO_MULTIPLE_DOTS
     char *dot; /* pointer to ifname extension, or NULL */
 #endif
+    int max_suffix_len = (z_len > 3 ? z_len : 3);
+    /* Leave enough room in ifname or ofname for suffix: */
+    if (strlen(iname) >= sizeof(ifname) - max_suffix_len) {
+        strncpy(ifname, iname, sizeof(ifname) - 1);
+   /* last byte of ifname is already zero and never overwritten */
+        error("file name too long");
+    }
     strcpy(ifname, iname);
     /* If input file exists, return OK. */
```

Fig. 3. The patch for `gzip`

Without going into too much detail, we can see that the patch checks the length of the variable `iname` and outputs an error message when it is too long. The patch adds a few statements in a small function `get_istat()`, which corresponds to only 7 additional lines (including two comment lines). The original source of `gzip.c` contains 1,744 lines, which means that the change accounts for roughly a 0.4% change in the source code (when not considering other dependencies). Although it is a very small change in the source code, we find that none of the 75 non-empty functions in the patched binary (which contains more than $8,500$ instructions) is syntactically the same as any functions in the unpatched binary. Further analysis shows that this is mainly due to basic block re-ordering.

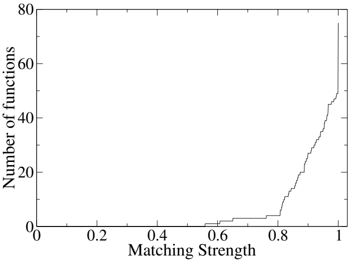


Fig. 4. Matching strengths of functions in `gzip` (CDF)

Although all non-empty functions are changed syntactically, BinHunt was able to find the correct matching for all non-empty functions. Figure 4 shows the number of non-empty functions that had matching strengths less or equal to the value on the x-axis. Despite the fact that all non-empty functions contain syntactic changes, BinHunt managed to find more than 10 function matches that have matching strengths of 1.0, which means that these matched functions contain basic blocks that are functionally equivalent and that use the same register allocation. There is also a large number of function matchings that have matching strengths between 0.9 and 1.0, which means that these matched functions contain basic blocks that are functionally equivalent, although some of these basic blocks use different register allocations. (See definitions of matching strength in Sect. 4.2 and Sect. 5.3.)

Among the matched functions with a matching strength less than 0.8, there was only one pair that differed substantially (by 26) in the number of unmatched basic blocks, which was function `treat_file()`. The rest of the matched functions had zero or a very small number of unmatched basic blocks. However, `treat_file()` is not the function `get_istat()` where changes were made in the source code, but rather its parent function, because function inlining was applied during compilation in both patched and unpatched versions. The parent function is a very large function with more than 900 basic blocks. BinHunt was able to find common subgraphs of a size almost the same as the unpatched function (all basic blocks were matched correctly except the two nodes between which new basic blocks are added), and therefore identified a few additional basic blocks that exist only in the CFG of the patched binary (see Fig. 5 for the additional basic blocks in the dotted rectangle).

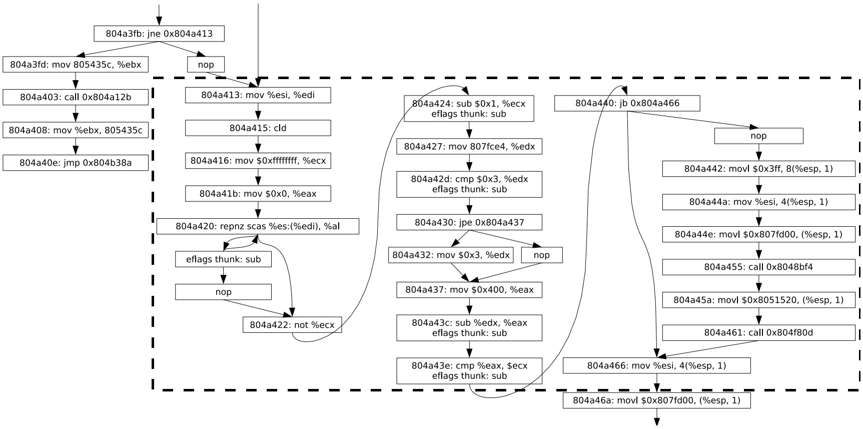


Fig. 5. Difference found in the parent function of `get_istat()`

Looking at the first few basic blocks within the dotted rectangle in Fig. 5, we found the assembly code as shown in Fig. 6. The `repnz` instruction repeats doing something as long as a byte is non-zero, which, in most cases, is used to find the terminating byte in a string. In this case study, it is used for the same purpose to find the length of the string at `edi`. By tracing the register `edi` we easily found that the string is actually the input parameter representing the file name.

```
mov %esi, %edi
cld
mov $0xfffffff, %ecx
mov $0x0, %eax
repnz scas %es:(%edi), %al
```

Fig. 6. Assembly code for differences found in `gzip`

In this case study, BinHunt identified a few basic blocks that correspond to changes in the source code of `gzip`. It took about an hour to finish the analysis on a desktop computer with a 2.1GHz CPU. The reason why it takes relatively long is because all functions in the patched version are syntactically different from functions in the unpatched version. In this case, we configured BinHunt to perform graph isomorphism on all permutations of the 75 functions in each binaries. This also means that the analysis time would not increase substantially if there were more semantic differences between the two binaries.

6.2 "Dot_dot" Vulnerability in tar

The patch for the buffer overflow vulnerability in gzip inserts additional instructions into a function, which result in a few additional basic blocks in a CFG. However, these additional basic blocks do not alter the control flow of the original program. BinHunt is very powerful in identifying this type of change in source code because our binary difference analysis is based on analysis of the control flow of the program. Although many patches share the same characteristics of the one for gzip, there are patches which change the control flow of the program. In this subsection, we describe an application of BinHunt to a program in which the patch changes the control flow of the original program.

```
bool contains_dot_dot (char const *name) {
  char const *p = name +
    FILE_SYSTEM_PREFIX_LEN (name);
  for (;; p++) {
    if (p[0]==’.’ && p[1]==’.’ &&
      (ISSLASH(p[2]) || !p[2]))
      return 1;
    do { if (! *p++) return 0; }
      while (! ISSLASH (*p));
  }
}
```

Unpatched

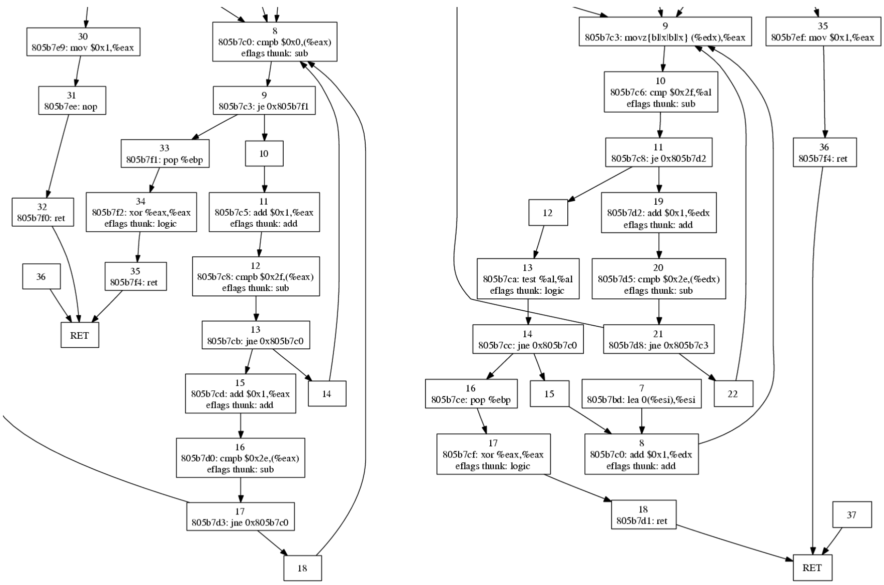
```
bool contains_dot_dot (char const *name) {
  char const *p = name +
    FILE_SYSTEM_PREFIX_LEN (name);
  for (;; p++) {
    if (p[0]==’.’ && p[1]==’.’ &&
      (ISSLASH(p[2]) || !p[2]))
      return 1;
    while (! ISSLASH (*p))
      { if (! *p++) return 0; }
  }
}
```

Patched

Fig. 7. The unpatched and patched functions in tar

A version of the program tar has an input validation error called the "dot_dot" function vulnerability. (See http://www.securityfocus.com/bid/25417/info for more information.) Attackers may exploit the vulnerability to overwrite files on the computer. Figure 7 shows the unpatched and patched versions of the function in which the vulnerability is found. The difference in the source code is that the unpatched function uses a do-while loop, whereas the patched function uses a while loop. This patch changes the control flow of the program. Again, BinHunt was able to find the correct matching between all non-empty functions from the two binary files. In this case, more than 95% of the matches had a matching strength of 1.0. This is due to the fact that function contains_dot_dot() is located towards the end of the binary file, and changes in it do not result in much basic block re-ordering. Out of the 470 non-empty functions, the match for function contains_dot_dot() had the smallest matching strength of 0.571622.

The graph isomorphism calculation on the CFGs for function contains_dot_dot() found a matching for about two-thirds of the basic blocks (the unpatched version has 36 basic blocks and the patched version has



(a) Unpatched

(b) Patched

Fig. 8. Part of the CFGs of function `contains_dot_dot()`

37 basic blocks). Figure 9 shows the number of basic blocks that were matched with a matching strength less than the value on the x-axis. The analysis of the unmatched basic blocks to identify the vulnerability required a little more effort because of the changes in control flow. Figure 8 shows part of the CFGs of the unpatched and patched function `contains_dot_dot()`.

As can be seen from Fig. 8, a loop exists in both CFGs, i.e., the path between basic block number 8 and 18 in the unpatched function and the path between basic block number 9 and number 8 in the patched function. There is a special basic block that does the comparison of a byte with the character "/" (ASCII `0x2f`); this is basic block 12 in the unpatched function and basic block 10 in the patched version. It can been seen from the CFGs that the comparison is performed in the middle of the loop in the unpatched function, and at the beginning of the loop in the patched function. With this, the vulnerability is found.

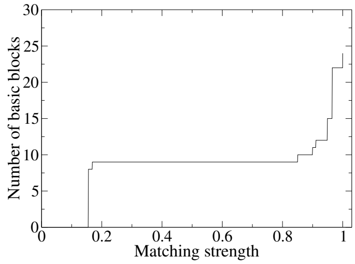
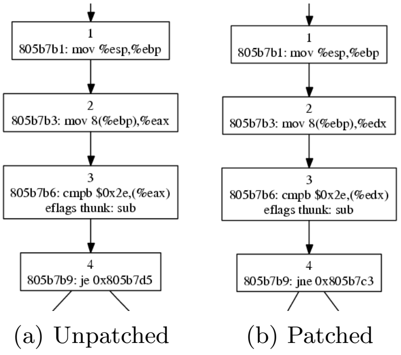


Fig. 9. Matching strengths of basic blocks in `contains_dot_dot()` (CDF)

In this case study, there are more than 41,000 instructions in each of the two binaries. It took about 30 minutes for BinHunt to finish the analysis.

One of the reasons why it took only 30 minutes is that some of the functions in the two binaries are exactly the same, and so BinHunt did not need to perform the graph isomorphism for these functions.



(a) Unpatched (b) Patched

Fig. 10. Different register allocation for function `contains_dot_dot()`

Another interesting thing to note is about different register allocation. Figure 10 shows the first few basic blocks of function `contains_dot_dot()` for both the unpatched and patched versions. We can see from basic block number 2 that `eax` is used in the unpatched function, while `edx` is used in the patched function for the same purpose. This is an example of syntactic differences that BinHunt managed to skip when finding semantic differences.

6.3 Application Folder Information Disclosure in `ASP.NET`

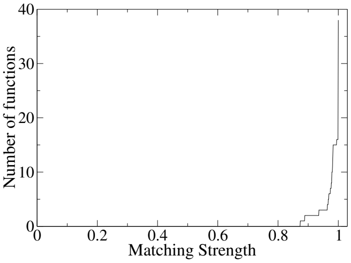


Fig. 11. Matching strengths of functions in `ASP.NET` (CDF)

The last case study we did was on Microsoft .NET framework 2.0 (`ASP.NET`). Unlike the previous two cases in which we compiled the source to obtain the binary executables independently, in this case study we downloaded the binary files directly from the software vendor. `ASP.NET` in many versions of Microsoft Windows allows remote attackers to bypass access restrictions via unspecified "URL paths" that can access Application Folder objects "explicitly by name" (CVE-2006-1300). This vulnerability occurs because `ASP.NET` only checks for slash ("/") and does not consider `%5c` (the ASCII code for "\") when checking for accessibility.

BinHunt found that there are 38 non-empty functions in the unpatched versions of the binary files and 39 non-empty functions in the patched version, and found the correct matching for 38 functions. Figure 11 shows the number of functions that were matched with a matching strength less than the value on the x-axis. We can see that all matched functions had a high matching strength.

In this case study, it was trivial to find the semantic difference as it corresponds to an unmatched function `FlipSlashes()`, which is called from function `HttpFilterProc()` to perform additional checks. BinHunt managed to locate the unmatched basic block in function `HttpFilterProc()` which corresponds to the call of `FlipSlashes()`. This case study shows that BinHunt works as expected on binary files downloaded directly from the software vendor.

7 Related Work

The structural comparison tools BinDiff [4] (and its extension [3]) and BindView (http://www.bindview.com/Services/Razor/Papers/2004/comparing_binaries.cfm) are most related to our work. These tools construct a maximal subgraph isomorphism between the sets of functions in two versions of the same executable file. There are two major distinctions between these systems and BinHunt.

First, BinHunt contributes a more thorough technique for identifying the maximum common subgraph isomorphism. BinDiff and BindView use a greedy method to extend a matching, and thus an erroneous match will propagate, leading to a failure to find the maximum subgraph isomorphism. In comparison, BinHunt uses a backtracking technique to find the maximum isomorphic subgraphs (see Sect. 5). While in general this would be exceedingly expensive, we develop optimizations to make it practical. Inaccurate matches added to the result will be replaced by better ones subsequently in the backtracking process.

Second, BinHunt uses a novel technique for basic block comparison using symbolic execution and theorem proving (see Sect. 4). This method can determine if two basic blocks are functionally equivalent, which overcomes the difficulty encountered when, e.g., basic blocks use different register allocations. In contrast, BinDiff uses heuristics to test if two graphs or basic blocks are similar. For example, BinDiff compares two graphs by calculating the number of basic blocks, edges and callers. BindView matches basic blocks based on instructions present in them. Due to the reliance on comparing actual instructions, a significant number of locations are falsely identified as changes [3].

There are also binary difference analysis tools to produce and apply patches (bsdiff, bspatch, xdelta, jdiff, jpatch, etc.). They capture all syntactic differences between binaries; as described previously, such differences may not correspond to semantic differences, and so they do not suffice for the goals of this paper.

8 Conclusion and Limitations

In this paper, we define the problem of finding semantic differences in binary executables, and introduce a novel technique BinHunt based on control flow analysis. When compared with previous techniques, BinHunt uses a more thorough graph isomorphism technique for identifying the maximum common induced subgraph isomorphism. Unlike previous techniques, BinHunt does not rely on many heuristics when finding the maximum common subgraph. BinHunt also makes use of a novel technique to compare the functionality of two basic blocks using symbolic execution and theorem proving. In case studies on different versions of three common programs, we showed that BinHunt is able to find the semantic differences with high accuracy.

A limitation of BinHunt is that its analysis efficiency drops when the number of semantic differences between binary files increases. This is due to the graph isomorphism technique that BinHunt uses. The backtracking algorithm works the best when the two graphs are similar to each other. In applications where the differences between the two binary files are large, a different graph isomorphism technique should be used. BinHunt does not work on packed code, either. We leave these topics for future work.

References

1. Balakrishnan, G., Gruian, R., Reps, T., Teitelbaum, T.: Codesurfer/x86 - a platform for analyzing x86 executables. In: Proceedings of the Conference on Compiler Construction (2005)
2. Brumley, D., Poosankam, P., Song, D., Zheng, J.: Automatic patch-based exploit generation is possible: techniques and implications. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy (May 2008) (to appear)
3. Dullien, T., Rolles, R.: Graph-based comparison of executable objects. In: Proceedings of SSTIC 2005 (2005)
4. Flake, H.: Structural comparison of executable objects. In: Proceedings of the GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment 2004 (2004)
5. Ganesh, V., Dill, D.: A decision procedure for bit-vectors and arrays. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590. Springer, Heidelberg (2007)
6. DataRescue Inc. IDA Pro, http://www.datarescue.com/idabase/
7. King, J.: Symbolic execution and program testing. Communications of the ACM 19(7) (1976)
8. Krissinel, E., Henrick, K.: Common subgraph isomorphism detection by backtracking search. Software — Practice and Experience 34 (2004)
9. Levi, G.: A note on the derivation of maximal common subgraphs of two directed or undirected graphs. Calcolo 9 (1972)
10. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassembly. In: Proceedings of the 11th ACM Conference on Computer & Communication Security (CCS 2003) (2003)
11. Raymond, J., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. Journal of Computer-Aided Molecular Design 16 (2002)
12. Sankoff, D., Kruskal, J.B.: Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison. Addison-Wesley Pu. Co., Reading (1983)
13. Ullman, J.: An algorithm for subgraph isomorphism. Journal of the Association of Computers and Machines 23 (1976)
14. Vintsyuk, T.K.: Speech discrimination by dynamic programming. Cybernetics and Systems Analysis 4(1) (1968)
15. Wang, Z., Pierce, K., McFarling, S.: Bmat - a binary matching tool for stale profile propagation. J. Instruction-Level Parallelism 2 (2000)

Enhancing Java ME Security Support with Resource Usage Monitoring

Alessandro Castrucci, Fabio Martinelli, Paolo Mori, and Francesco Roperti

Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche
via Moruzzi,1 - 56124 Pisa (Italy)
{alessandro.castrucci,fabio.martinelli,paolo.mori}@iit.cnr.it

Abstract. Both the spreading and the capabilities of mobile devices have dramatically increased over the last years. Nowadays, many mobile devices are able to run Java applications, that can create Internet connections, send SMS messages, and perform other expensive or dangerous operations on the mobile device. Hence, an adequate security support is required to meet the needs of this new and evolving scenario.

This paper proposes an approach to enhance the security support of Java Micro Edition, based on the monitoring of the usage of mobile device resources performed by MIDlets. A process algebra based language is used to define the security policy and a reference monitor based architecture is exploited to monitor the resource usage. The paper also presents the implementation of a prototype running on a real mobile device, along with some preliminary performance evaluation.

1 Introduction

In these last years, the market of mobile devices, such as mobile phones or Personal Digital Assistants (PDAs), has grown significantly. The computational power and the capabilities of mobile devices have increased too. For example, modern mobile devices are able to connect to Internet, to read and write e-mails, and also to run Java applications.

Java Micro Edition (Java ME) is a version of the Java platform for mobile and embedded devices. Java ME for mobile devices mainly consists of two components: the *Mobile Information Device Profile* (MIDP) and the *Connection Limited Devices Configuration* (CLDC). The security model provided by Java ME is not flexible enough to allow the spreading of Java ME applications, MIDlets, developed by third party companies, because it takes into account the provider of the MIDlet only, i.e. the principal that signed the MIDlet. If the MIDlet provider is trusted according to the list of trusted principals stored on the device, the MIDlet is allowed to perform any security relevant action. Instead, MIDlets that come from unknown providers are not allowed to perform security relevant actions, and the mobile device user is tediously prompted to explicitly allow each of them. To avoid to be asked again, the user could choose to allow any further invocation, thereby disabling any further control on the MIDlet.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 256–266, 2008.

This paper proposes an approach to enhance the Java ME security support based on the continuous monitoring of the usage of the mobile device resources. The usage of the resources of the mobile device is defined in terms of the sequences of actions that a MIDlet performs on them, i.e. the behavior of the MIDlet. The resource usage monitoring we define is fine-grained, history-based and continuous. The monitoring is *fine-grained* because we define a set of security-relevant actions, i.e. actions performed on the mobile device resources that could be critical for the device security, and we monitor all the invocations to such actions performed by MIDlets. The monitoring is also *history-based*, because to decide whether a MIDlet is allowed to perform a given action, we take into account the sequence of all the actions that have been executed by the MIDlet itself since it was started. This implies that the existence of the right to perform an action is not static, because it depends on the actions that have been previously executed. Furthermore, the monitoring is *continuous* because our approach allows to define conditions that are continuously (repeatedly) checked, and as soon as a condition is violated, a system action is executed, such as interrupting the MIDlet, even if it is not executing a security relevant action. The security policy describes the allowed resource usage patterns in terms of sequences of actions that MIDlets are allowed to perform, and the conditions that should be satisfied before the execution of each action, after, or continuously. The security policy is stored on the mobile device and is applied to each MIDlet that is executed.

Hence, with respect to Java ME security, the main novelties of the proposed security support are that: *i)* the rights granted to a MIDlet to access mobile device resources do not depend on the MIDlet provider; *ii)* these rights are not static; *iii)* the monitoring of the resource usage is continuous, i.e. the controls are not executed before the access only, but also while the access is in progress, and, consequently, a MIDlet could be interrupted while running, even if it is not performing a security relevant action.

1.1 Paper Structure

The paper is structured as follows. Section 2 describes the standard Java ME security support, and reports previous attempts to enhance it. Section 3 describes our approach to improve the security of the Java ME architecture by monitoring the MIDlet execution. In particular, Section 3.1 describes the security policy we adopted, and Section 3.2 describes the architecture of our framework. Section 4 describes the implementation of a prototype running on a real mobile device, the HTC Universal smart-phone, along with a preliminary performance evaluation. Section 5 draws the conclusion.

2 Related Work

The Java ME security support involves all the basic components of the Java ME architecture: Mobile Information Device Profile (MIDP), Connected Limited Device Configuration (CLDC) and Kilobyte Virtual Machine (KVM). The security support provided by CLDC [17] concerns the low level and the application level security. The low level security regards the KVM, and guarantees that MIDlets do not harm the device while running. The application level security, instead, deals with security relevant operations performed by MIDlets, such as accesses to libraries or resources. To execute MIDlets, CLDC adopts a sandbox that ensures that: the MIDlet must be pre-verified; the MIDlet cannot bypass or alter standard class loading mechanisms of the KVM; only a predefined set of APIs is available to the MIDlet; the MIDlet can only load classes from the archive it comes from; and, finally, the classes of the system packages cannot be overridden or modified.

The security support provided by MIDP [7,8] defines four protection domains: *Untrusted, Trusted, Minimum, and Maximum.* Each MIDlet is bound to one protection domain depending on its provider, and this determines the value of its permissions. Permissions refer to the actions that the MIDlet can perform during its execution and their value can be either *allowed* or *user.* For example, the `javax.microedition.io.Connector.http` permission refers to HTTP connections. If the value of this permission is *allowed*, then the permission is granted, otherwise, if the value is *user*, a user interaction to explicitly grant the right is required every time that the MIDlet tries to establish an HTTP connection. When asked by the MIDP security support, the user can deny the right to execute the action, or can allow it by choosing among three possible values: *oneshot*, *session*, *blanket*. If the user chooses *oneshot*, the right to execute the current action is granted, but the user will be asked when the MIDlet will try to perform the same action again. If the user chooses *session*, the right to execute the current action is granted to the MIDlet until it terminates. Instead, if the user chooses *blanket*, the MIDlet will be allowed to perform the action until it is uninstalled or the permission is explicitly changed by the user. In other words, this disables any further control on this action.

A security study of Java ME has been presented by Kolsi and Virtanen in [9], where they described the possible threats and the security needs in a mobile environment. In particular, they described how MIDP 2.0 solved some security issues of MIDP 1.1, but they concluded that some problem are still present. A security analysis of Java ME has been presented also by Debbabi et al. in [2], [3] and [4]. In these papers, they detail the MIDP and CLDC security architecture, and they identify a set of vulnerabilities of this architecture. Moreover, they also test some attack scenarios on actual mobile phones. However, the previous papers do not propose any improvement to the Java ME security support to solve the security issues they described.

An attempt of extending the Java ME architecture with an enhanced security support is shown in [6]. This paper proposes a runtime monitor architecture that consists of a *Runtime Monitor*, a *Policy Manager* and a *History Keeper*. The Runtime Monitor is in charge of making resource access decisions, and relies on the Policy Manager to identify the relevant application-specific policy. Once the policy is identified, the Runtime Monitor evaluates its conditions in conjunction with resource usage history information of the system and MIDlet, as obtained from the History Keeper. This architecture enforces policies written in *Security Policy Language* (SPL) [16]. SPL is a constraint based security policy language that allows to express simultaneously several types of authorization policies, hence allowing the definition of complex access control models (e.g. RBAC, DAC, TRBAC).

The main difference with the approach proposed in this paper is in the kind of security controls that are performed. Our approach is focused on the continuous monitoring of the mobile device resources usage, and the security policy defines the rights of a MIDlet by describing the resource usage patterns that the MIDlet is allowed to perform. These patterns could be very complex, and are expressed through a process algebra based language. Moreover, our approach defines continuous controls, that consist of conditions that are continuously checked while the MIDlet execution is in progress. When one of these conditions is violated, the monitor executes a control action, such as interrupting the MIDlet execution. This requires a more complex support then the one for enforcing access control policies.

3 Runtime Monitoring

This paper proposes to enhance the security support of Java ME by monitoring the usage of the resources of the mobile device. This implies the monitoring of the execution of MIDlets to intercept the security relevant actions that they perform on the mobile device resources and the enforcing of a security policy that defines the admitted patterns of these actions.

The actions that are considered as security relevant are the ones that allow the MIDlet to interact with the underlying resources, such as establishing a network connection, sending an SMS message, initiating a phone call, and so on. Hence, we identified a set of methods of the MIDP and CLDC core classes that implement the security relevant actions, and we monitor the execution of these methods. For example, `javax.microedition.io.Connector.open(url)` is the method that creates a connection with the entity represented by `url`, and the method `javax.wireless.messaging.MessageConnection.send(msg)` interacts with the mobile device to send an SMS message to a remote device.

3.1 Security Policy

This section gives a short description of the language for defining security policies. We adopt an operational policy language because we believe that it is closer to user's expertise than denotational ones. Since we deal with sequences of actions, we use a *PO*licy *L*anguage based on *P*rocess *A*lgebra (*POLPA*) (see also [1,11,10]). A policy results from the composition of security relevant actions, control actions, predicates and variable assignments, as described by the following grammar:

$$P ::= \bot \parallel \top \parallel \alpha(\boldsymbol{x}).P \parallel c.P \parallel p(\boldsymbol{x}).P \parallel \boldsymbol{x} := \boldsymbol{e}.P \parallel P_1 or P_2 \parallel P_1 par_{\alpha_1,..,\alpha_n} P_2 \parallel \\ \{P\} \parallel Z$$

where P is a policy, $\alpha(\boldsymbol{x})$ is a security relevant action, c is a control action, $p(\boldsymbol{x})$ is a predicate, $\boldsymbol{x}$ are variables and Z is a constant process definition $Z \doteq P$. The difference between security relevant actions and control actions is that security relevant actions are the ones that the MIDlet tries to perform on the mobile device resources, while control actions are executed by our monitoring support to enforce the security policy. Interrupting and suspending the MIDlet execution are two examples of control actions. The informal semantics is the following:

- $\bot$ is the *deny-All* operator;
- $\top$ is the *allow-All* operator;
- $\alpha(\boldsymbol{x}).P$ is the *sequential operator for security relevant actions*, and represents the possibility of performing an action $\alpha(\boldsymbol{x})$ and then behave as P;
- $c.P$ is the *sequential operator for control actions*, and represents the possibility of performing a control action c and then behave as P;
- $p(\boldsymbol{x}).P$ is the *sequential operator for predicates* and behaves as P in the case the predicate $p(\boldsymbol{x})$ is true;
- $\boldsymbol{x} := \boldsymbol{e}.P$ assigns to variables $\boldsymbol{x}$ the values of the expressions $\boldsymbol{e}$ and then behaves as P
- $P_1 or P_2$ is the *alternative operator*, and represents the non deterministic choice between P_1 and P_2;
- $P_1 par_{\alpha_1,\ldots,\alpha_n} P_2$ is the *synchronous parallel operator*. It expresses that both P_1 and P_2 policies must be simultaneously satisfied. This is used when the two policies deal with actions (in $\alpha_1, \ldots, \alpha_n$);
- $\{P\}$ is the *atomic evaluation*, and represents the fact that P is evaluated in an atomic manner. P here is assumed only to have one action, predicates and assignments;
- Z is the constant process. We assume that there is a specification for the process $Z \doteq P$ and Z behaves as P.

As usual for (process) description languages, derived operators may be defined. For instance, $P_1 par P_2$ is the *parallel operator*, and represents the interleaved execution of P_1 and P_2. It is used when the policies P_1 and P_2 deal with disjoint actions. The policy sequence operator $P_1; P_2$ may be implemented using the policy languages operators (and control variables) (e.g., see [5]). It allows to put two process behaviors in sequence. By using the constant definition, the sequence and the parallel operators, the iteration and replication operators, `i(`P`)` and `r(`P`)` resp., can be derived. Informally, `i(`P`)` behaves as the iteration of P zero or more times, while `r(`P`)` is the parallel composition of the same process an unbounded number of times.

Many different execution patterns may be described exploiting POLPA. Figure 1 shows a simple example of security policy to avoid redirections to other web sites while accessing a predefined web site, “www.siteA.it“. At the beginning of the execution, this policy allows the MIDlet to open any network connection. However, if the MIDlet opens a HTTP or a HTTPS connection with the predefined site, then it cannot open any other connection with any other site. On the other hand, if the MIDlet opens a connection with an URL that is not the predefined one, then in this session it cannot open this site anymore.

Moreover, the policy does not allow the MIDlet to open any connection to the predefined site if the protocol is not HTTP or HTTPS. For instance, this policy could be adopted when executing MIDlets that implement Internet browsers, such as Opera Mini [14], to avoid redirections to malicious sites when accessing the predefined site.

```
r([(address(url)!=“www.siteA.it“) ].javax.microedition.io.Connector.open(url))
or
r([( (protocol(url)==HTTP) or (protocol(url)==HTTPS) ) and
   (address(url)==“www.siteA.it“) ].javax.microedition.io.Connector.open(url))
```

Fig. 1. Example of security policy

Figure 2 shows another example of POLPA policy. In this case, the policy allows the MIDlet to send no more than 10 SMS messages to italian users only (i.e. if the telephone number begins with “+39“). As a matter of fact, the policy allows the MIDlet to execute the `javax.microedition.io.Connector.open` method only if the protocol is the SMS one and if the telephone number begins with “+39*“, and it allows the MIDlet to invoke for 10 times only the method `javax.wireless.messaging.MessageConnection.send`.

```
N:=0.
i([((protocol(url)==SMS) and (address(url)==“+39*“)].
  javax.microedition.io.Connector.open(url)
  or
  ( [(N<10)].javax.wireless.messaging.MessageConnection.send(msg).
    N:=N+1)
)
```

Fig. 2. Example of security policy

Figure 3 shows a further example of POLPA policy where the MIDlet is allowed to open a network connection with the site “`http://www.siteA.it`“, and then, either it opens a network connection with the site “`http://www.siteB.it`“ within 10 seconds, or it is interrupted by the control action `revoke_execution`. As a matter of fact, the control action `revoke_execution` is executed as soon as the predicate `[(timer > 10)]` is satisfied. In this example we suppose that the variable `timer` represents a timer.

```
[(url == “http://www.siteA.it“)].javax.microedition.io.Connector.open(url).
timer:=0.
( [(timer > 10)].revoke_execution()
  or
  [(timer ≤ 10) and (url == “http://www.siteB.it“)].
  javax.microedition.io.Connector.open(url)
)
```

Fig. 3. Example of security policy

3.2 Runtime Monitor Architecture

The architecture for the runtime monitoring follows the *reference monitor* model, and consists of two main components: a Policy Decision Point (PDP) and a Policy Enforcement Point (PEP), as shown in Figure 4.

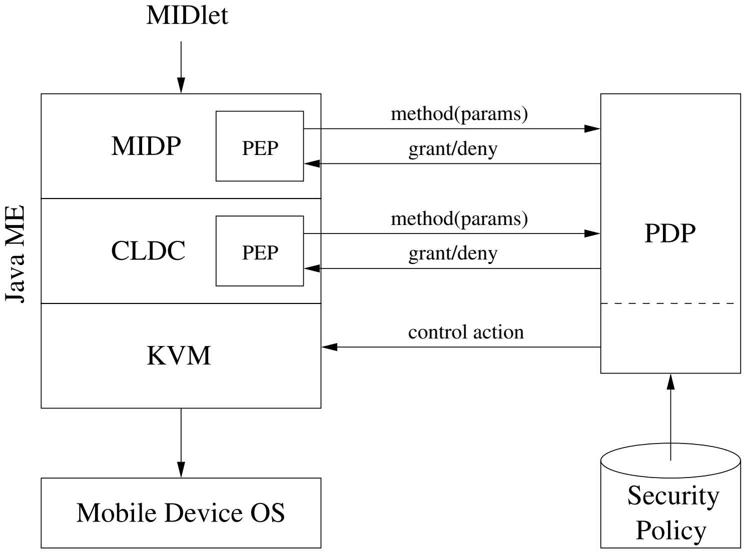


Fig. 4. Runtime monitoring architecture

The PEP is integrated in the MIDP and CLDC components of the Java ME architecture, while the PDP is implemented as a distinct component. This solution requires the modification of the Java ME architecture to embed the PEP, while does not require any modification of the MIDlets, hence allowing the execution of standard MIDlets.

The PEP has two main tasks during the execution of a MIDlet: *i)* intercepting the security relevant methods invocation, and *ii)* enforcing the decision resulting from the evaluation of the security policy on this method. When a security relevant method is intercepted, the PEP invokes the PDP, by passing it the method name and all the invocation parameters. To embed the PEP in the MIDP and CLDC methods we modified the source code of those methods by inserting the invocation of the PDP at the beginning and at the end of the method code. In this way the policy is evaluated and enforced both before and after the execution of the method. The PEP also enforces the decision of the PDP. If the PDP decision is positive, the execution of the method is permitted, then the PEP continues the execution of the original method code. Instead, if the result is negative, the execution of the method is denied, and the PEP terminates it by throwing a Java Exception. In this case, if the PDP invocation has been made before the execution of the method, the method execution is skipped.

The PDP is the component that decides whether a given security relevant method can be performed in a given state according to the security policy. The PDP is initiated by the KVM before beginning the execution of the MIDlet bytecode. The PDP initially gets the security policy from the local storage, and builds an internal representation of the policy. This internal representation is used to efficiently evaluate the policy against the security relevant actions that the MIDlet tries to perform. The PDP consists of two parts: a passive one and an active one. The passive PDP is invoked by the PEP for each security relevant method that the MIDlet tries to execute, before and after the execution of the method. When the policy evaluation process terminates, the passive PDP returns the decision to the PEP that enforces it. The active PDP, instead, repeatedly tests whether a control action should be executed, by evaluating the predicates before the active control actions. A control action is active when the previous actions in the sequence defined by the policy have been already executed by the MIDlet. For example, in the security policy shown in Figure 3, the `revoke_execution` control action is active only after that the connection to the site "`http://www.siteA.it`" has been established. The passive PDP, for each security relevant action executed by the MIDlet, updates the set of active control actions.

4 Implementation

We developed a prototype of the modified Java ME runtime environment that runs on a real mobile device, a HTC Universal smart-phone, exploiting the PhoneME Feature Software MR2 [15]. The PhoneME feature software is an implementation of the main components of the Java ME architecture, such as the MIDP v2.0, the CLDC v1.1, the *Wireless Message API* and many others. The PhoneME Feature Software MR2 release includes the full source code. In particular, the KVM code is developed in C++, both for efficiency reasons and because it interacts with the underlying operating system. The code of the Java ME core classes is developed partly in Java and partly in C or C++. In this case too, C functions are used mainly to implement the interactions with the underlying operating system. Our customized version of PhoneME was built on a desktop computer exploiting the OpenEmbedded development environment [12] and configuring the cross compiler for the specific mobile device architecture. The PhoneME was installed on a HTC Universal smartphone (also known as QTEK 9000) running Linux (Openmoko distribution [13]).

The PEP and the PDP have been integrated in the PhoneME source code, according to the architecture described in Figure 4. From the implementation point of view, the Policy Decision Point consists of two threads developed in C language mainly for efficiency reasons. The PDP is started by the KVM before the execution of the MIDlet bytecode. Once activated, one PDP thread suspends itself waiting for an invocation from the PEP component, while the other repeatedly check the predicates paired with the active control actions every t seconds, where t is a system parameter. If one of these predicate is violated, this thread enforces the corresponding control action through the native AMS support provided by the phoneME.

The PEP, in contrast, consists of a Java class and a C function. The Java class includes a method, checkPolicy, to activate the PDP. The invocations to the checkPolicy method are embedded in the source code of the Java ME methods that implement the security relevant actions, before and after the original code. In this way, the security policy is checked before and after the execution of the security relevant action. The PEP communicates with the PDP exploiting shared variables and semaphores. The enforcement of the PDP decision, when the right to execute an action has been forbidden, is implemented by throwing a SecurityException error in the code of the Java ME method. This error will be reported to the MIDlet.

4.1 Experimental Results

This section evaluates the impact of our enhanced security support on the performances of the Java ME Virtual Machine. As a matter of fact, the MIDlet monitoring slows down the execution of the MIDlet because of the time spent to check the security policy. The overhead on the execution time depends on the enforced policy. As a matter of fact, in general, complex security policies take more time to be evaluated than simple ones. Moreover, the performance degradation also depends on the specific MIDlet, i.e. on the methods it invokes. In particular, the overhead depends on the number of security relevant methods invoked by the MIDlet with respect to the invocations to other methods, because the security relevant methods are the ones that introduce the overhead.

The MIDlet used for our tests performs 10 HTTP connections to a remote site. This is the worst case from the performance point of view, because most of the methods invoked by this MIDlet are security relevant ones, and introduce the monitoring overhead. In a real case MIDlet we expect that the most of the methods invoked are not security relevant ones and, consequently, the overhead due to our security support will be less relevant. To perform these tests the MIDP permissions support has been disabled.

Figure 5 shows the execution times of the chosen benchmark. The three experiments have been executed with the original phoneME software, and with the phoneME software instrumented with our enhanced security support enforcing

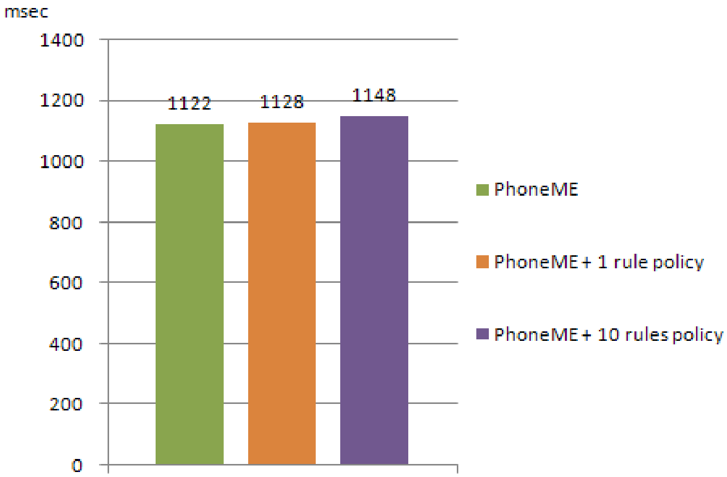


Fig. 5. Performance evaluation

two policies, one with one rule only and the other with 10 rules. In the policy with 10 rules, we chose a worst case again, because the policy has been written in a way such that the PDP must examine all the rules before finding the one that allows the method invoked by the MIDlet. The execution time of the test MIDlet executed on the original PhoneME environment is compared against the one of the same MIDlet executed using PhoneME with our enhanced security support. The results show that the overhead introduced by our system is small. In fact, the enforcing of a policy with one rule results in a 0,5% overhead, while the enforcing of a 10 rules policy results in an overhead of 2,6%. As previously discussed, this results represent the worst case, and we think that in case of a real MIDlet the overhead will be even less.

5 Conclusion and Future Work

We proposed an approach to enhance the security support of the Java ME architecture based on the monitoring of the behavior of the MIDlets. The experiments we carried out on the prototype we developed showed that the overhead due to security controls is very low. Hence, we think that this approach can be successfully adopted on modern mobile devices to allow the secure execution of MIDlets.

References

1. Baiardi, F., Martinelli, F., Mori, P., Vaccarelli, A.: Improving grid service security with fine grain policies. In: Meersman, R., Tari, Z., Corsaro, A. (eds.) OTM-WS 2004. LNCS, vol. 3292, pp. 123–134. Springer, Heidelberg (2004)

2. Debbabi, M., Saleh, M., Talhi, C., Zhioua, S.: Java for mobile devices: A security study. In: ACSAC 2005, pp. 235–244. IEEE Computer Society, Los Alamitos (2005)
3. Debbabi, M., Saleh, M., Talhi, C., Zhioua, S.: Security analysis of mobile java. In: Proceedings of the Sixteenth International Workshop on Database and Expert Systems Applications, 2005, pp. 231–235. IEEE Computer Society, Los Alamitos (2005)
4. Debbabi, M., Saleh, M., Talhi, C., Zhioua, S.: Security evaluation of J2ME CLDC embedded java platform. Journal of Object Technology 2(5), 125–154 (2006)
5. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM 21(8), 666–677 (1978)
6. Ion, I., Dragovic, B., Crispo, B.: Extending the java virtual machine to enforce fine-grained security policies in mobile devices. In: Choi, L., Paek, Y., Cho, S. (eds.) ACSAC 2007. LNCS, vol. 4697. Springer, Heidelberg (2007)
7. JSR 118 Expert Group. Mobile information device profile for Java 2 micro edition. Java Standards Process JSP 118 (November 2002), http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html
8. JSR 118 Expert Group. Security for gsm/umts compliant devices recommended practice. addendum to the mobile information device profile. Java standards process (November 2002), http://www.jcp.org/aboutJava/communityprocess/maintenance/jsr118/
9. Kolsi, O., Virtanen, T.: Midp 2.0 security enhancements. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences 2004(2004)
10. Martinelli, F., Mori, P., Vaccarelli, A.: Towards continuous usage control on grid computational services. In: Proc. of International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services 2005, p. 82. IEEE Computer Society, Los Alamitos (2005)
11. Martinelli, F., Mori, P.: A model for usage control in grid systems. In: Proceedings of GRID-STP. IEEE Press, Los Alamitos (2007)
12. Openembedded project, http://www.openembedded.org
13. OpenMoko project, http://openmoko.org
14. Opera Mini, http://www.operamini.com
15. phoneME project. phoneME Feature Software Milestone Release 2, http://phoneme.dev.java.net
16. Riberio, C., Guedes, P.: An access control language for security policies with complex contraints. In: Proceedings of Network and Distributed System Security Symphosium (NDSS 2001) (2001)
17. Sun Microsystems Inc. The connectected limited device configuration specification. Java Standards Process JSR 139 (March 2003), http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html

Pseudo-randomness Inside Web Browsers

Zhi Guan, Long Zhang, Zhong Chen, and Xianghao Nan

Institute of Software, School of EECS, Peking University.
Key Lab of High Confidence Software Technologies (Peking Univ.),
Ministry of Education
{guanzhi,zhanglong,chen,nanxh}@infosec.pku.edu.cn

Abstract. With the increasing concerns over the security and privacy of Web based applications, many solutions based on strong cryptography have been proposed to protect client side Web applications against attacks such as phishing, pharming and even server side attacks. While strong cryptography is used, one critical building block in cryptosystem, the random number generator, is often neglected. Considering this situation, in this paper we design and implement a pseudo-random number generator only rely on ubiquitous Web browser abilities - JavaScript, HTML and AJAX. We also provide a mechanism called *Pseudo-cookie* for JavaScript programs to access operating system services for retrieving random or entropy values without changing Web browser security policies. The security model, analysis and performance evaluation demonstrate that our method is secure and efficient.

1 Introduction

With the increasing popularity of the Web 2.0 applications such as Google Gmail, Google Docs and Flickr, current Web browsers, not only act as the interface for static web page browsing, but also change into a platform for data outsourcing, information sharing and collaborations among group members, together with the services behind the World Wide Web. This new computing paradigm has become very successful during the last few years and has led Web based application a better replacement of corresponding traditional desktop program. In spite of the easy of use these web based applications have provided, the security and privacy of organizations and individual users are more prone to be threatened than the traditional desktop counterparts. One of the reasons, from technical point of view, is that as a platform the Web browser lacks of enough client side security mechanisms such as strong cryptography and secure storage than the operating system. Some solutions, such as BeamAuth [1], WebIBC [2] and ClipperZ [3] have been proposed to bridge the gap between the limited Web browser capabilities and the security requirements based on introducing strong cryptography into client side Web applications.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 267–278, 2008.

Although these cryptography systems made great reinforcement to the security of Web based applications through exploiting strong cryptography mechanisms such as symmetric encryption, MAC (Message Authentication Code), public key cryptography and Identity-Based Encryption (IBE), as an important building block of the proposed cryptosystems, the random number generator (RNG) is often neglected by the designers. A strong random number generator for cryptographic utilization is simply assumed to be exist and available. Unfortunately, it is not the fact. The truth is no secure random number generator is available for Web based applications.

Random number generator is one of the most fundamental primitives in cryptography that has been researched for many years. "*A random number generator is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits*" [4]. True randomness is widely used in cryptography applications, such as symmetric and asymmetric cryptography key generation. Weak random numbers may offer the adversary abilities to bypass the hardness of breaking a cryptosystem. However, in spite of the importance of random number generation security, many designs, standards and protocols used in practice instead leave the random number generator to non-security exports, many real world implementations only rely their security on insecure solutions. The most recent example is a random number generator defect found in Debian Linux [5]. This flaw results in a large amount of security applications include SSH, OpenVPN, SSL/TLS, DNSSEC and X.509 tools into the danger of easily broken. Therefore, current Web based security applications even without a random number generator will result in great danger.

In this paper we describe the design and implementation of a random number generator for Web based security applications. The security of random number generation in Web browsers is discussed and particular threats are analyzed. Through accumulating entropy from the browser, the user interactive operations and local environment variables, we present a secure random number generator completely through ubiquitous Web browser capabilities such as HTTP, JavaScript, AJAX [6]. We also introduce a new mechanism called *Pseudo-cookie* for JavaScript programs to access operating system services without changing the Web browser security policies, we exploit the method to retrieve randomness and use it to seed and refresh the state of our generator, which can largely improve the performance of the generator. The security analysis and performance evaluation show promising values for real world applications. As we know, this is the first work addressing the security of random number generation in a pure Web environment.

The rest of this paper is organized as follows: in Section 2 we introduce the background of random number generator together with related attacks, followed by the security model in Section 3, then we will introduce the design, implementation and performance evaluation of our web based random number generator. In session 4, we will propose the *Pseudo-cookie*, a mechanism to bridge the gap between browser and local environment. At last the paper will be concluded in Section 5.

2 Random Number Generator

2.1 Theory and Practice

Linear Congruential Generator. While widely available, the mathematic random function `rand()`[1] in glibc and `Math.random()` [7] in JavaScript are not feasible for cryptography utilization. The algorithms implemented in `Math.random()` in Safari 3 and Firefox 2 (we get this information through read the code[2],[3], we assume other browsers might be the same) are called *linear congruential generator*, which produce a sequence of numbers $x_1, x_2, \ldots$ according to the linear recurrence $x_n = ax_{n-1} + b \texttt{ mod } m, n \geq 1$; integers a, b and m are parameters of the generator and x_0 is the seed. Although this generator provides the uniform distribution random numbers, it does not satisfy the unpredictable requirement. Given a partial output sequence, the remainder of the sequence can be reconstructed even if the parameters a, b and m are unknown.

True Random Number Generator (TRNG). A TRNG requires a naturally occurring source of randomness such as unpredictable physically procedures. The implementation is through an especial hardware device of software program to collect randomness from precise timing of hardware events to monitoring people behaviors.

Pseudo-random Number Generator (PRNG). While true randomness is widely available in the nature, it's hard for deterministic computing system to provide true random number generators through deterministic algorithms. Instead the pseudo-random number generator is used, which extends a short truly random number sequence to a much longer sequence that "appears" to be random. The input to the PRNG is called the *seed*, while the outputs of the PRNG are called *pseudo-random numbers*.

Random Number Generators in Practice. For the rarity of true randomness, the output of TRNG is often used as the input of PRNG. Many software random number generators have been proposed, implemented and researched. In a chapter in [8] a detailed survey of software random number generators are discussed, in [9] a generalized software architecture is introduced, in [10] the model of secure random number generation service is discussed, and [11] and [12] are analyses of random number generation on Windows and Linux operating system. As a good engineering practice, pseudo-random number generators have been provided as a system service by modern operating systems. For it is more feasible for OS to collect entropy from hardware events and user inputs. Unix-like operating system implements kernel level pseudo-random number generator and provide the interface through a virtual device `/dev/random`, while Windows provided similar API to provide random numbers. Different from the random devices

[1] http://www.gnu.org/software/libc/libc.html

[2] The WebKit Open Source Project. http://nightly.webkit.org/

[3] Mozilla Developer Center, http://developer.mozilla.org/en/docs/Download_Mozilla_Source_Code

above is a device implemented in kernel space. Windows RNG is most implemented in user space, so that the design and implementation of Windows can not resist forward security attack, which is considered a big flaw [11].

2.2 Break Cryptography through Weak Randomness

Random number generator is an important building block for cryptography and used in many security applications, such as symmetric key generation, initial vector selection in symmetric encryption, public key generation, nonce in protocols, and random public key algorithms. The security of these systems is based on the condition that the random number is unpredictable. Here we give some examples on how to break a Web based security application through the weak randomness in use.

ECDSA (Elliptic Curve Digital Signature Algorithm) [13] is the signature scheme utilized in [2], which is a variant of DSA on elliptic curve groups. With the same security level, ECDSA can provide shorter key length and signature size than RSA. Our attacks on ECDSA are through predictable random values. Given the elliptic curve parameters are $T = (p, a, b, G, n)$, in which p specifying the prime field $\mathbb{F}_p$, $a, b \in \mathbb{F}_p$ specifying the elliptic curve equation $E(\mathbb{F}_p) : y^2 = x^3 + ax + b \texttt{ mod } p$, G is a base point on curve $E(\mathbb{F}_p)$ and n is the order of G. Signer's private key is integer $d \in_R [1, n-1]$, public key is elliptic curve point $Q = d \cdot G = G + G + \ldots + G$, addition of base point G with d times. When signing on message m, the signer must randomly select an integer $k \in_R [1, n-1]$, compute $R = k \cdot G = (x_R, y_R)$, $r = x_R \texttt{ mod } n$ and $s = k^{-1}(H(m) + dr) \texttt{ mod } n$, H is a cryptographic hash function and (r, s) is the ECDSA signature. If k is predictable, the attacker can extract signer's private key by $d = \frac{s \cdot k - H(m)}{r} \texttt{ mod } n$. Even if the attacker can not predict the value k but know the signer to use the same random integer k on signing two different message m_1 and m_2 with result (r, s_1) and (r, s_2) respectively, the attacker can also calculate the random number k through $k = \frac{H(m_1) - H(m_2)}{s_1 - s_2} \texttt{ mod } n$, and get signer's private key through $d = \frac{s \cdot k - H(m_1)}{r} \texttt{ mod } n$. Similar attacks also available for other cryptosystems, such as [14].

While the original papers do not mention what generators are in use, we assume that it is the `Math.random()` default provided by JavaScript, which is a linear congruential generator in both Firefox and Safari. An efficient attack on DSS (Digital Signature Standard) through linear congruential generator is introduced in [15] which can be easily converted into attack against ECDSA. For the length of this paper we do not provide the details here.

3 Threat Model

A random number generator for Web based applications must be secure against different threatens from both inside and outside attackers of the browser. One type of typical attackers is the network eavesdropper that can get all the traffic between the Web server and the browser; Another type of attack is local malware like Trojan horse which can eavesdrop not only all the network traffic, but also local communication between browser and the operating system, the local malware can even manipulate some data from the operating system to the browser. Local attackers and remote attackers sometimes can also learn some inner variables through memory probing or pharming techniques. The reason of this attack considered to be short period is threefold: these attacks often can only get a snapshot of system; The session of a Web application is short because web pages will be closed after browsing, and the close of both the attack page or the target application page means the end of this attack; and the last, if this attack can last, then its power is overwhelming and no solutions can be used to protect the random number generation except for the reinforcement of the whole system. The Web service provider should also be considered as an attacker because when a Web application is designed to provide privacy for data outsourcing, and then sever side will also be classified into potential attackers. However, the server is considered not to provide a backdoor in its application. This is because for Web applications the source code can be easily reviewed by everyone and mechanisms such as JavaScript code signing [16] introduced by Netscape and XML signature [17] can provide authentication for the application from security experts.

From the above discussion, formally speaking an attacker to the Web based random number generator would have the following capabilities:

- Have all the design and implementation details of the generator.
- Prompting the generator for output random number and observing this output.
- Observing and even influencing some of the data that is used to refresh the internal state of the generator.
- Learn the internal state of the generator at will, but this attack only last for a short period.

Compared with the attacker in desktop environment, our Web based threat model is weaker according to the attacker capability on tampering the generator internal state. The rationality of this difference is that with the security policies, a Web browser should be seen as a secure environment, and protecting a JavaScript program's inner data should be seen as the duty of the browser. So we will not mix up a random number generator's model with that of a browser.

Same to desktop random number generators, a secure Web based generator should satisfy the above requirements:

- Pseudo-randomness. The generator's output seems random to an outside observer.
- Forward security. An adversary who learns the internal state of the generator at a specific time cannot learn anything about previous outputs of the generator.
- Backward security/break-in recovery. An adversary who learns the state of the generator at a specific time does not learn anything about future outputs of the generator, provided that sufficient entropy is used to refresh the generator's state.

4 Design and Implementation

4.1 System Construction

Our construction is modified from the Barak-Halevi model [10] with some variants for the characteristic of Web based applications. In traditional desktop environment, the security applications can be divided into the random number provider and the random number consumer. The consumer, while in web applications a JavaScript programs in a single page must implement either, on the other side, the JavaScript program in a page often need less random values than the desktop counterpart.

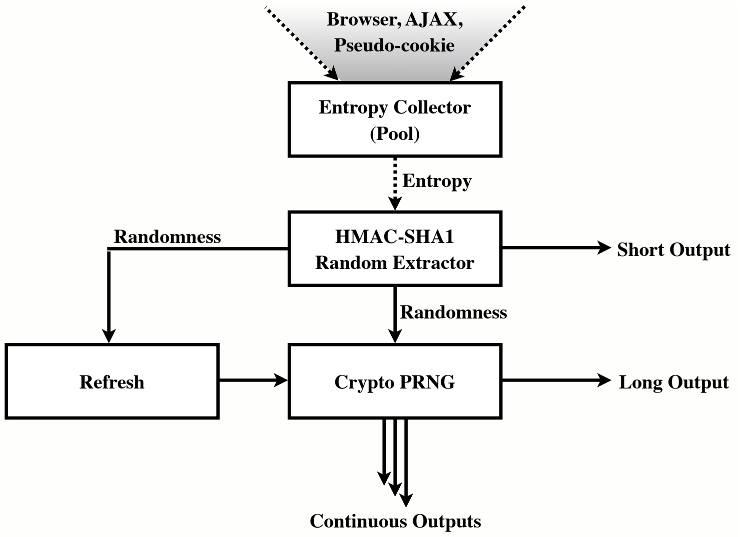


Fig. 1. Construction of the Web Based Random Number Generator

In our construction shown in Fig. 1, the generator includes four components: the entropy collector, the random extractor, the cryptography pseudo-random number generator and a refresh algorithm.

- The entropy collector accumulates entropy from the browser events, user inputs, remote server or other resources.
- The randomness extraction function $\texttt{extract}(e) \rightarrow s$ that converts the high-entropy but non-uniform distributed input e to a shorter but uniform distributed output s.
- The cryptographic PRNG function $\texttt{prng}(s, m) \rightarrow r$ that generates m bytes of pseudo-random values expanded from the seed s.
- The refresh function $\texttt{refresh}(s, x) \rightarrow s'$ that refreshes the current seed by additional entropy generated by the function extract.

Unlike desktop applications to consume random numbers from a system random generator as the provider, a random number generator inside a web page acts as both the randomness provider and the consumer. If the extracted entropy generated from the random extractor is enough (typically 128 bytes or 160 bytes) for the application, these values will be used directly by the application. Considered if the application only consumes very short random numbers, then the randomness can all extracted from the output of random extractor. Considered if the entropy is not enough, the cryptographic pseudo-random number generator will be called to expand the extracted randomness to enough length and output. If the session of the Web application will last for a long time and require continuous new keys, for example in the online Web based chatting, our construction will provide the full function, as the Barak-Halevi model, every time new random number generated, the inner state of cryptographic PRNG will be renewed, and when a fixed period of time, if enough entropy is collected the refresh algorithm will mix new randomness with current inner state to refresh into a new state.

Entropy Collection. The entropy for our generator is coming from three kinds of sources, the browser environment, the browser events including user interface events and network events, user inputs textual entropy feeding, and server feeding.

When a page is loaded, the entropy accumulation callback function is registered to the browser environment. When some events with high entropy are occurred, the accumulator will receive the event and the occurred time. The following events are the main entropy resources.

- The current window place and size: 2 bytes.
- `onkeypress` event is occurred when a keyboard key is pressed,
- `onmousemove` event is occurred when the mouse is moved, and the x and y coordinates (4 bytes) will be added into the random pool. The user interface inside a page will notice the user to move his mouse randomly for entropy collection.
- `onmouseover` is occurred when the mouse is moved over and element in an HTML document (2 bytes). Web page includes some invisible elements on the page for the moving mouse to trigger this event frequently.

Browser events are not the only sources of entropy, our construction also provide other facilities. With the AJAX technique, the JavaScript program can request a block of random data from the server through a `Microsoft.XMLHTTP` ActiveX JavaScript object in IE or `XMLHttpRequest` JavaScript object in other browsers without reloading the current page. While this is much easier and faster, for the security of the system, random data from server should not exceed that from local browser events.

User input entropy can also be utilized, for example, when user is asked to input some random texts, the key value, press time will be added into the random pool.

4.2 Randomness Consumption

How much randomness does a JavaScript cryptosystem consume inside a web browser? This depends on what kind of application or cryptosystem is utilized. Here we consider some typical applications:

Symmetric encryption with predefined key requires a random initial vector (IV) with length same as the block size of the block cipher, typically 128 bits, such as AES.

For password based encryption, the key derive function such as PBKDF2 in PKCS #5 standard [18] requires excessive random bytes as salt value, the default length is 8 bytes, i.e. 64 bits. If the integration is needed, the generation of password based MAC requires other 64 bits of salt. Password based encryption established on symmetric encryption requires extra keys and IV, totally 320 or 384 bits.

For single message protected by public key cryptography, for example, an PKCS #7 cryptographic message syntax [19] to envelop a message with encryption and digital signature. For public key encryption, ECIES(Elliptic Curve Integrate Encryption Scheme) [13] and ECDSA both require a random k with 192 bits length for encryption and signature generation, together with the 256 bits symmetric encryption key and IV random values, a PKCS #7 package require about 640 bits of random values.

Randomness Extraction. In the area of random extractors one such example is the so-called "strongly universal" or "pairwise independent universal" hash functions, the other kind is extractors from CBC, Cascade and HMAC from block ciphers and cryptographic hash algorithms. For security and engineering considerations we apply HMAC as our random extractor instead of also available AES CBC-MAC and SHA-1 hashing. In our benchmark, for the typical hash function SHA-1 is nearly double speed of the typical block cipher AES with 128 bits key length in nearly all environment. While simply hashing is considered not secure enough in the research result of [20]. To generate uniform distribution random values from collected entropies, the input min-entropy size should be at least double size of the output. Which means if the application require n-bit random values and the random pool has at least $2n$ min-entropy values, then the pseudo-random generation procedure can be bypass, and only the extracted values is enough for the utilization of cryptographic applications.

Randomness Estimation. So the question becomes, how to estimate the min-entropy of collected entropies. While this is an even harder problem in the area of random number generation, with different environment the randomness will be different, and the amount of entropy can even be manipulate by the adversary. So we accept the suggestion of [10], giving up estimating the randomness, just collect as much entropy as possible. In our implementation, the randomness extraction will not start until the random pool is filled with 512 bytes entropy. The min-entropy of these 4096 bits data will definitely fulfill the 160×2 bits (HMAC-SHA1 output size) requirement. This procedure requires about 10 seconds.

Generation and Refreshing. Our pseudo-random number generation and refreshing scheme is similar to that of [10]. The difference is that our `pseu` function can generate a longer sequence of random values, not just double size of the inner state. Our `pseu` can be easily implement by any secure pseudo-random number generators. In our implementation, the standard FIPS 186 generator [21] with SHA-1 is choose for the SHA-1 code can be reused in the generator.

4.3 Performance

The main time consuming of our generator comes from cryptography computation and entropy collection. We run a primitive JavaScript benchmark program for the evaluation of our generator with different options. We test the performance of AES with 128 bits key length and SHA-1 hashing algorithm on a Laptop with 1.83 GHz 2-Core CPU running Mac OS X.

Table 1. JavaScript Cryptography Benchmark (Bytes per second)

	SHA-1	AES 128
Firefox 3	140	65
Safari 3	104	51
Firefox 2	41	15
Opera 9	23	8
OpenSSL	111,739	38,774

As shown in 1, on two new browsers, Safari 3 and Firefox 3 (RC1), the block cipher speed is about 50 KBytes per second while the hashing algorithm speed is doubled. These algorithms are fast enough for randomness extraction and pseudo-random number generation even they are about 1000 times slower than OpenSSL, the compiled binary counterpart.

As a comparison, the entropy collection speed from browser events is no more than 50 bytes per second, more than 1000 times slower than the cryptographic operations! Which means the entropy collection procedure is the performance bottle neck of this system.

5 Pseudo-cookie

With the benchmark in last section it is known that, while the cryptographic computation is efficient, our Web based generator spend most of the time on entropy collection instead of cryptography computations when no other entropy source is provided. Compared to the limited entropy resources in Web browser, operating system has much more entropy resources with higher quality. Unfortunately, in Web browser security model the downloaded JavaScript programs can only run inside a restricted "sandbox" that isolates them from the rest of the operating system. Scripts are only permitted to access data in the current page or closely related pages (generally those from the same site through the same protocol). This is called *the Same-Origin Policy.* So no access is granted to the local devices, events and programming interfaces for the collection of entropy or retrieving randomness.

The common technique for extending browser capabilities is browser plug-in, a software module can interact with the browser and provide specific functions. For example, a dedicated plug-in can retrieve random numbers or collecting entropy from operating system and provide these values to JavaScript programs in the browser. However, this mechanism does not fulfill the security requirements of Web application security. From the security point of view, a plug-in break *the Same-Origin Policy* of Web browser, and unlike JavaScript programs with all the source code opened, a compiled plug-in can do evil without any notice, while the user have no choice but to trust it completely. In practice, the deployment requires to develop different plug-ins for every browser in various operating systems, and need every user to install the correct plug-in into his browser, which is a huge burden for both vendors and customers.

A question is, might the Web applications exploit services from local operating system without breaking the Web browser security policies? The answer is yes, here we introduce a new mechanism named *Pseudo-cookie* to bridge the gap between the JavaScript programs in web browsers and the services provided by local operating system. In Web browser a regular HTTP cookie is a piece of data include name-value pairs stored on local disk that can be created, read and written by JavaScript programs. While a pseudo-cookie is nearly all the same for scripts and slightly logic difference for browsers. The difference is twofold:

- For a script, the pseudo-cookie is nearly the same as a regular cookie, only with a pre-defined name-value pair with the fixed name "random". A script can read values from the "random" name-value pair (create and write operations are also permitted but nonsense for random number generation), every time the script reads, the return value will be a different random value.
- For the browser when a cookie reading operation is evoked, if the target is the "random" name-value pair browser should assume the corresponding value has been changed by another programs , so the browser must retrieve this value from reading the cookie file again then return the newest value. The browser need not do this extra work for any other cookie values. The implementation has two choices; the browser can do this all by itself. When the random cookie reading triggered, the browser should retrieve random value from system and return this value, or the browser can just assume the cookie has been changed by other programs since its last access, and must read the cookie file again to return the newest value. A separated filesystem driver or a hood program can be used to replace the ordinary cookie file.

Our implementation of pseudo-cookie is based on WebKit, an open source browser engine adopted by Safari, KDE desktop environment in Linux. The random value is directly read from `/dev/random` device by the modified engine. The appended C code is very short, no more than 100 lines. Although we did not mention and implement in our prototype, this mechanism can be easily extended as a generalized interface to support any other services.

6 Conclusion and Future Work

We have presented a Web based random number generator, a critical cryptographic building block for Web based applications. Through the security model we give an analysis of threats and security requirements of random number generator in Web environments. Our design and implementation can be well adapted by Web based security applications without any browser plug-ins or breaking current security policies. To solving the lack of entropy in Web browser, we also propose a mechanism call pseudo-cookie to bridge a gap between operating system service and the Web application by exploiting the feature of cookies in modern browsers. We still remain some interesting topics such as the entropy estimation and extension of the pseudo-cookie paradigm that require future research.

References

1. Adida, B.: Beamauth: two-factor web authentication with a bookmark. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 48–57. ACM, New York (2007)
2. Guan, Z., Cao, Z., Zhao, X., Chen, R., Chen, Z., Nan, X.: WebIBC: Identity Based Cryptography for the Client Side Security of Web Based Applications. In: Proceedings of ICDCS (2008)
3. Barulli, M., Solaroli, G.C.: Clipperz: the free and anonymous online password manager (2007)
4. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
5. Debian Security Advisory: DSA-1571-1 openssl – predictable random number generator (2008), http://www.debian.org/security/2008/dsa-1571
6. Wenz, C.: JavaScript und AJAX. Galileo Computing (2007)
7. ECMA: Standard ECMA-262, ECMAScript Language Specification 3rd (1999), http://www.ecma-international.org/publications/standards/Ecma-262.htm
8. Gutmann, P.: The design and verification of a cryptographic security architecture. submitted thesis (2000), http://www.cs.auckland.ac.nz/pgut001/pubs/thesis.html
9. Gutmann, P.: Software generation of practically strong random numbers. In: Proceeding of 7th USENIS Security Symposium (1998)
10. Barak, B., Halevi, S.: A model and architecture for pseudo-random generation with applications to /dev/random. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM Conference on Computer and Communications Security, pp. 203–212. ACM, New York (2005)
11. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the windows random number generator. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 476–485. ACM, New York (2007)
12. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the linux random number generator. In: S&P, pp. 371–385. IEEE Computer Society, Los Alamitos (2006)
13. Hankerson, D., Menezes, A., Vanstone, S.: Guide to elliptic curve cryptography. Springer, Heidelberg (2004)

14. Zheng, Y., Matsumoto, T.: Breaking Real-World Implementations of Cryptosystems by Manipulating their Random Number Generation. In: Proceedings of the 1997 Symposium on Cryptography and Informations Security (1997)
15. Bellare, M., Goldwasser, S., Micciancio, D.: Pseudo-random number generation within cryptographic algorithms: The dds case. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 277–291. Springer, Heidelberg (1997)
16. Mozilla.org: Signed Scripts in Mozilla (2007), http://www.mozilla.org/projects/security/components/signed-scripts.html
17. W3C: W3C Recommendation on XML-Signature Syntax and Processing (2002), http://www.w3.org/TR/xmldsig-core/
18. RSA Laboratary: PKCS5: Password-Based Cryptography Standard version 2.0 (1999), ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs-5v2-0a1.pdf
19. RSA Laboratary: PKCS7: Cryptographic Message Syntax Standard version 1.6 (1997), ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-7/pkcs-7v16.pdf
20. 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)
21. FIPS 186: Digital Signature Standard. FIPS Publication 186, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia (1994)

Verifiable and Revocable Expression of Consent to Processing of Aggregated Personal Data

Henrich C. Pöhls

University of Passau, Institute of IT-Security and Security Law (ISL), IT-Security
Innstr. 43, 94032 Passau, Germany
henrich.poehls@uni-passau.de

Abstract. We have identified the following three problems for the processing of aggregated personal information with respect to privacy preferences: Unverifiable proof of consent, unverifiable proof of consent for aggregated personal data, and no verification if the consent is still established. We constructed a solution based on a hash tree structure and digitally signed only the hash tree's root value. Thus, a verifiable signature can be retained even if data items are omitted and a valid signature serves as signal of consent. To re-assure that no change of consent has taken place we propose the use of certificate revocation mechanisms. As a side-effect these mechanisms allow to maintain a record of personal data usage and thus creates a win-win situation for both parties involved.

1 Introduction

Looking at the automatic processing of aggregated data the question of privacy shall always be raised. Following the definition given by the EU [5] data items that are personally identifying will be called *personal data* and "shall mean any information relating to an identified or identifiable natural person (*data subject*);" [5]. To be legally allowed to process this personal data the data subject needs to express his consent to the processing: "the *data subject's consent* shall mean any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed." [5]

In digital systems this consent can be expressed by the data subject's election to "opt-in" or "opt-out" [10] of certain processing. This consent is usually expressed at the time the user provides the service with his personal data. Services collecting personal data state in their *privacy policies* how the service will handle the submitted personal data. As these services process data we call them *processors*. The processor's privacy policy can be machine understandable, like P3P [21] for websites. Most of the time we find services constructed in such a way that users not consenting to the service's privacy policy will not be able to participate[1]. Thus, a direct relationship between the data subject and the data

[1] Often found: "By using the service you consent to the collection and use of information about you in the ways described in this Privacy Policy."

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 279–293, 2008.

processor exists. A direct relationship is the simplest case and transmits the data subject's expression of consent directly to the processor.

Besides the processor's privacy policy the processing of personal data can be restricted by the data subject. The restrictions and constraints under which the data subject allows the processing will be called *privacy preferences*. Most of the time the service's privacy policy is equal to the data subject's privacy preferences. The processor's privacy policy can be seen as a suggestion for privacy preferences. The data subject accepts them as his own by expressing consent. Hence, the processor dictates the restrictions instead vice versa. We do not further elaborate on this problem, but the proposed solution allows the data subject to set forth his privacy preferences.

In case of a dispute, the processor is obliged to present a proof that consent was established, as "personal data may be processed only if the data subject has unambiguously given his consent" [5]. With no verifiable paper-trail, i.e. no paper lottery ticket with a ticked "opt-in" box, the service retains no direct proof of established consent. Implicitly, the service's sign-up process and the privacy policies the data subject agreed to, in order to use the service, at the time of data submission can be used. In order to serve as a proof that consent was established, this process has to be documented, time stamped, and attested by a trusted third-party. Even in the simple case of a direct relationship, constructing a verifiable proof of the data subject's consent to data processing does not exist in most environments and requires trusted third-parties.

In a loosely connected, decentralized environment, with web services, agents, or Web 2.0 [17], a direct relationship is often not existing or its establishment would be a hindrance to the user. With compound services, like most web services, the original processor engages additional sub-services to carry out the processing. Each sub-service, if respecting privacy, would request a proof of established consent before processing the given personal data. With the advent of semantic applications on the semantic web [1] the exchange of such data will increase. Services will combine or re-combine data items from different sources and construct new data sets, a process we denote by *aggregation*. An aggregator should, under circumstances set fourth by the data subject, be able to derive the consent for the processing of aggregated personal data from the initial consent established for the source data.

Personal data is also exchanged between services who have no direct relationship to the data subject. On the other hand, the data subject can rectify, erase or block the processing [5]. All these actions will *vanish the consent*. Until the time personal data is processed, the data subject might have vanished his consent to processing, rendering any further processing "unconsented" and thus inducing possible legal challenges if processed. At most processing services, that have a direct relationship to the data subject, are aware of this state change in the consent from *established* to *vanished*. However, a service with no direct relationship is left unaware that the data subject's consent has vanished. Vanishing consent removes the service's ability to rightfully process and use the personal data held by it any longer. A suggested approach is to time limit a given consent, but most often the time of state change can not be predetermined, i.e. a change of address. Thus, services need a way to automatically determine that the consent is still established.

To summarize, we have identified the following three problems:
No possibility for third-parties to verify:

(i) data subject's consent to personal data processing,
(ii) data subject's consent to aggregated personal data, and
(iii) that the data subject's consent is still established.

Our approach presented in this paper solves these three problems by retaining a proof of established consent with the data. This proof is verifiable by third-parties as well as by the data subject, and shows what privacy policy and combination of data items the data subject consented to. Additionally, a once established consent can be vanished, to indicate that the processing once consented to is no longer consented to.

The rest of the paper is organised as follows: Starting with a short scenario in Sect. 2, we will define more terms and clearly outline the problems in Sect. 3. We will present our solution in Sect. 4 and show how existing mechanisms from public key infrastructure (PKI) can be used to technically implement the concept. We thoroughly discuss our solution in Sect. 5, including a discussion on the performance and a discussion of existing work. We also show the security of our solution in the presence of an attacker. Finally, we conclude in Sect. 6.

2 Scenario

In the given scenario, Mr. Luxe, is assumed as the data subject. Mr. Luxe has a profile on a social network service (SNS), in the scenario called *Soc*. *Soc* is the service Mr. Luxe has a direct relationship with, and he has consented that *Soc* uses his personal data like email and relations as set forth by *Soc*'s privacy policy. Now *Soc* is no walled garden and data items, like the list of Mr. Luxe's friends Alice and Bob, are accessible by sub-services[2]. Apart from keeping his social relations on *Soc*, Mr. Luxe is a registered user on the website *example.org*. *example.org* allows registered users to upload examples, but Mr. Luxe only seldomly visits it. However, his friend Alice is an active user of *example.org* and regularly uploads new examples.

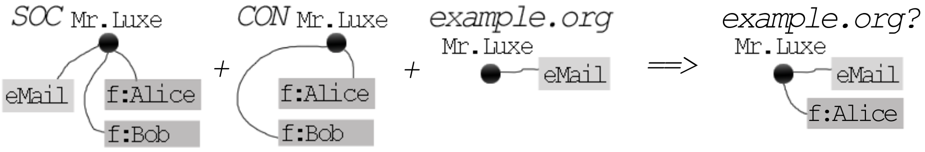


Fig. 1. Scenario data sets representing Mr. Luxe

[2] In SNS terminology often called "apps","widgets" or "gadgets".

A sub-service, called *Con*, now offers the functionality to indicate which known friends are also registered members on a website[3]. To enhance the user's experience, *example.org* incorporates *Con*'s service to offer additional notifications whenever friends are active on the website, for example sent an email when new content is added by a friend. From the website's point of view both, *Con* and *Soc*, are third-parties. The website knows its list of registered users, Mr. Luxe and Alice are both in this list, but *example.org* does not know their social relations. Now through the use of *Con*, *example.org* is given the information that Alice is a friend of Mr. Luxe on *Soc*. From the registration and during the use of *example.org*, the website has collected its own set of information about Mr. Luxe and Alice, for example the email address and their browsing habits.

Some questions that arise are: Has Mr. Luxe consented to an aggregation of *example.org*'s data with data received from *Con*? Can *example.org* sent him emails whenever Alice posts something on the site, to boost his visits? Can *Con*, as a service in the middle, be presented with a consent to process and transfer the data that Mr. Luxe and Alice are friends without having a direct relationship with them? In the rest of the paper we will look more closely at the identified problems and present our solution that allows to answer the questions.

3 Semantical Data, Privacy Preferences vs. Policies, and the Identified Problems

The scenario indicates the need to flexibly allow the use of different portions of the personal data D. So, we split the personal data into atomic elements that comprise D, called *data items*, denoted as d_i. Each data item covers one semantic concept and can not be split further, i.e. a field in a data base. Hence, D is defined as a set of all d_i. Every single data-item is identifiable by a name $\{d_i\}_{ID}$ that carries the semantic meaning. The data item's value is denoted $\{d_i\}_{VAL}$. A data graph [19] captures the data items and the respective privacy preferences. The privacy preferences that are applied to a data item d_i are denoted as ppref_i. Each ppref_i can contain one or more privacy preference expressions $pref$ to express the privacy preferences for d_i. All the data subjects privacy preferences for D are denoted as PPref_D being the set of all the ppref_i.

How the privacy preferences expressions $pref$ and privacy policies $PPol$ are codified and later enforced is not within our scope, the only constraint is that it needs to be interoperable among the involved parties. Thus, we assume that the involved parties share a common vocabulary for the variable names as well as for the privacy preferences. So variables with the same semantic meaning can be identified, and the terms expressing the privacy constraints are understood by all parties. Ontologies can be used to establish this common vocabulary among all the participating parties. Hence, we assume that for variables and preferences a single ontology, including domain-dependent and domain-independent ontologies, exists [19].

[3] This sounds familiar to a service called Friend Connect [6], but they differ technically and policy-wise.

A boolean function *match*(*PPref,PPol*) allows to check if the data subject's preferences are fulfilled by a service's policy, for example by applying policy subsumption as described by Squicciarini et al. [19]. With this function a service can check which available third-party service's policy would comply, thus being able to hand personal data to complying sub-services only. Here the first problem becomes visible: The sub-service that receives the personal data lacks the ability to verify the data subject actually expressed his consent to process this data. We will now look at each problem in more detail.

3.1 No Verification of Consent to Data Processing

As mentioned, the sub-service is a third-party. Since it has no direct relationship with the data subject, it has no way of being provided with the user's consent through a sign-up process. The function *match* only allows to check if the provided privacy preferences can be fulfilled. Our goal is to ensure that the sub-service will still be able to verifiable check the data subject's consent before processing.

The proposed approach provides services with an additional boolean function *verify*. A positive result of $\mathit{verify}(S, D, \mathit{PPref}_D, \mathit{PPol})$ allows to have confidence that processing the data items of D following the privacy policy PPref_D is consented to by the data subject S. Vice versa, a negative result would indicate that the processing is not be consented to. With *verify* we free the sub-service from the need to trust the upper layer service, the service he got the data from. For a processor, regardless on what layer, it shall not matter if it has gained the data through a direct relationship, gained the personal data by transfer, or by other means like harvesting openly available sources. In all cases a positive result of *verify* assures the processor that processing under its *PPol* is consented to by the data subject.

3.2 No Verification of Consent to Aggregated Data Processing

The second problem comes from the aggregation of personal data items by a service that received them through different sources. First we also want to allow the opposite of aggregation to happen: Fragmentation. Fragmentation of data takes place rather often, as only some portions, some data items, of the complete data set are shared. From a privacy point of view sharing of a minimal sub-set of data items is preferred. The verification of a fragment given a data set A from the data subject S works as follows: If $\mathit{verify}(S, A, \mathit{PPref}, \mathit{PPol})$ is positive then $\mathit{verify}(S, C, \mathit{PPref}, \mathit{PPol})$ shall also yield a positive result iff $C \subseteq A$.

The opposite is the aggregation of two data sub-sets. For example the original and complete data set contains three data items $A = \{a_1, a_2, a_3\}$. Two sub-sets $\{a_1, a_2\}$, $\{\mathit{ppref}_1, \mathit{ppref}_2\}$ and $\{a_1, a_3\}, \{\mathit{ppref}_1, \mathit{ppref}_3\}$ are aggregated. As the aggregated data set will contain all data items, the verification of the aggregated set $\mathit{verify}(S, \{a_1, a_2, a_3\}, \{\mathit{ppref}_1, \mathit{ppref}_2, \mathit{ppref}_3\}, \mathit{PPol})$ yields the same verification outcome as $\mathit{verify}(S, A, \mathit{PPref}_A, \mathit{PPol})$. More details follow in Sect. 4.

3.3 No Verification of Consent State Changes

The last problem we identified was that the state of the consent expressed for the processing can change over time. Data fragmentation, aggregation, and the involvement of sub-services lead to the distribution of personal data among different services. Thus, re-use of personal data will occur over time. On the other hand, data items move virtually away from the data subject. Hence, each service needs the ability to gain re-assurance that an initially given consent is still established.

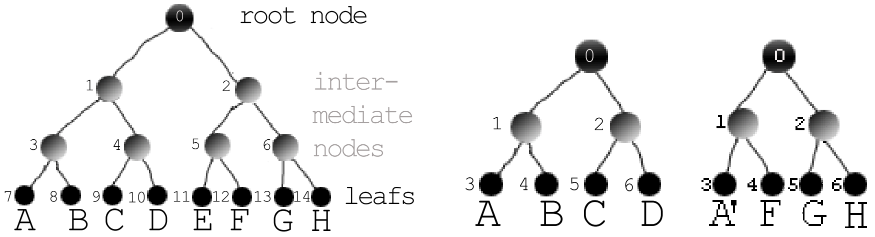
Three states are possible: *No consent*, *established consent*, and *vanished consent*. Two state changes are expected: From no expression to established, and from established to vanished. To initially establish consent the verifiable proof of consent, together with the data subject's privacy preferences is added and distributed along with the data set. To vanish consent it is not enough to just remove or strip the proof of consent completely. A retained expression of once established consent must serve as a proof for all processing occurred previous to the state change. Therefore, the two states established consent and vanished consent are verifiable, while the state no consent can not be verified. As data subjects are often unaware of their personal data' flows, each service shall be responsible to check that the consent has not vanished at processing time. This check must be carried out during the evaluation of the *verify* function. Of course, a status check involves additional communication between the service and the data subject or a trusted-third party acting on his behalf. This is comparable to a certificate revocation checking mechanism [8], and we will indeed propose to use a solution for handling the consent state changes technically based on certificate revocation.

4 Solution

We will now explain our proposed solution allowing to retain a verifiable expression of established consent, that can be later vanished, in more detail. To represent the data D, we choose a tree. Each leaf contains a tuple of a single data item d_i and its privacy preferences, so $\langle d_i, ppref_i \rangle$. For this tree we compute a hash tree, comparable to Merkle [13] hash tree. Starting from the leafs containing the tuples, each intermediate node is a hash of the concatenation of its children nodes. Finally the root node will be associated a hash value depending on: All $\langle d_i, ppref_i \rangle$, the tree's structure, and the association of tuples to leafs. We will then apply a digital signature scheme to just sign the root hash. We assume a public key to be bound to a data subject S, thus we are able to associate S with S's signature, for example by a trusted public key certificate. This protects the integrity and offers non repudiation with respect to identifying the key-pair used for signing. The leaf's and the intermediate node's hash values are then discarded.

In difference to a classical hash over $D||PPref$ a tree-based hash allows omission of sub-trees. To omit a sub-tree rooted at a given node, while still being able to re-compute the same root hash as in a tree without omissions, the node's hash value needs to be supplied as substitution for the omitted sub-tree. For example the leafs A and B in Fig. 2(b) can be removed if the hash of node 1 is given, we call this a *substitution hash* of 1. A signature over a tree rooted at node X is denoted by $\textit{signed}(X)$. So additionally to the tree's structure the nodes within the tree need to be identifiable and the association of tuples needs to be communicated. We will assume that this information is known. For brevity and better readability we use a shorthand notation: To indicate that only the data items A and B of the tree from Fig. 2(a) are present we write $A, B; 2, 4 + \textit{signed}(0)$, instead of $(7 : A), (8 : B); (2 : \textit{substitution hash of 2}), (4 : \textit{substitution hash of 4}) + \textit{signed}(0) + \textit{treeinfo}$.

We will call this data tree, accompanied by the digital signature and additional information, a *signed data tree*.



(a) One data tree with 8 leafs

(b) Two smaller trees

Fig. 2. Example binary trees, each leaf represents a *tuple* $\langle d_i, \textit{ppref}_i \rangle$

4.1 Retaining a Verifiable Expression of Consent

Our signal to indicate consent is the digital signature and verifying it works as follows: First the *verify* algorithm recomputes the hash tree root, using supplied substitution hashes where necessary. Note, we strictly forbid substitution hashes on paths leading to a not omitted tuple as this introduces ambiguities. So $A, B, C; 2, 4 + \textit{signed}(0)$ would not yield a positive verification. To indicate the omission of D one needs to supply $A, B, C, h(D); 2 + \textit{signed}(0)$. If the root hash can not be constructed verification fails. Second the digital signature is verified as usual according to the digital signature scheme. A positive signature verification for a tree indicates the consent to use any combination of the data items obeying the data item's privacy preferences that are part of the tree. So, with appropriate substitution hashes omissions are consented to.
Example: When node 2 in Fig. 2(a) is signed then any combination of the tuples E, F, G, and H are consented to by the signer.

4.2 Allowing Aggregation, While Retaining a Verifiable Expression of Consent

Verifying the aggregation of two verifiable signed data trees is only sensible, if the signers are the same. You can of course aggregate arbitrary data items, but a verifiable signature means that the data subject has consented to this aggregation. In other words, trying to verify the aggregation of verifiable data items from Alice with verifiable data items from Bob means asking if Bob consented to the aggregation with items of Alice and vice versa. As Bob does not need to know Alice, this aggregation of verifiable data items only makes sense if they are signed by the same key or the same identity.

We denote aggregation by $\otimes$. *Example from Fig.2(a):* E,G;$h(F)$,$h(H)$ + $signed(2) \otimes F = E, F, G; h(H) + signed(2)$ obviously verifies. Aggregation is especially obviously consented to when we can find a "smaller" singed data tree with a hash-value equal to a substitution hash in a "bigger" signed hash tree. *Example from Fig. 2(a):* We assumed A, B, C, and D being of equal value in the trees. Aggregating a big, partially known, but root signed tree with a smaller, incomplete signed sub-tree: $E, F, G; h(H), 1 + signed(0) \otimes A, B, D; h(C) + signed(1) = A, B, D, E, F, G; h(C), h(H) + signed(0)$. This result will verify, thus an aggregator, still without knowing the real values of C and H, is able to reproduce a verifiable signature of a larger tree after the aggregation.

An aggregator does not need a signed data tree to add an omitted data item d_x. If he knows the $ppref_x$ he can check prior to aggregation if $hash(\langle d_x, ppref_x \rangle)$ corresponds to a substitution hash. If it corresponds, he can add the tuple during an aggregation and *verify* will yield a positive result. Note, aggregating will only yield a positive verification if the data items added during aggregation directly match a substitution hash or if the aggregator is able to provide missing substitution hashes. *Example from Fig.2(a):* To verifiable aggregate $A, B; 4, 2 + signed(0)$ and C the aggregator needs $h(D)$.

A data owner, wanting to allow future aggregation of already known data items, does not need to release all the data item's values $\{d_p\}_{VAL}$. Instead, he can release $hash(\{d_p\}_{VAL})$, $\{d_p\}_{ID}$, and $ppref_p$. Doing so he signals his consent to possible future addition of the omitted values under the already stated privacy preferences.

Example from Fig.2(a): E,F;$h(G)$,$h(H) + signed(2)$ allows to later add G or H. In contrast E,F;$6 + signed(2)$ only verifies if either G and H, or their substitution hashes can be provided. ence, on signature generation the data subject can willingly express his consent to foreseen future aggregation, by supplying substitution hashes of leafs as place holders.

As the signature protects the integrity, aggregation with changed data items or changed privacy preferences will never yield a verifiable signature.

*Example from Fig. 2(b):*We assume the same signer for both trees. The aggregator's goal is to combine data items from the left tree A, B with data items from the right tree A', E. If $A = A'$, and the aggregator is presented with the inputs $B, C, D; h(A) + signed(0)$ and $A', E + signed(1)$, the aggregation $A', B, C, D + signed(0)$ will verify. All other combinations, for example B, E will not have a path to a common signed node, and will not verify. So to not consent to a combination, the data subject simply puts these data items into separate trees.

Example: To give no consent to the aggregation $(B, C, D) \otimes (E, F, G)$ while still allowing both of them to be combined with A results in the two signed trees depicted in Fig. 2(b) with $A = A'$.

Thus, to forbid certain aggregation the data subject needs to make sure that these data items do not end up in the same signed tree. Generating a different signed data tree for each different service the data subject interacts is quite natural. Note, our approach does not demand that a data item that is added during a consented aggregation must initially come from a signed data tree.
Example: Having $B, C, D; h(A) + \mathit{signed}(0)$, A can be added without respect to its origins. This assumes that the aggregator knows A's value, but also the identifier $\{A\}_{ID}$, and the privacy preferences ppref_A.

4.3 Allowing Status Changes of Consent by Using PKI Mechanisms

To enable a status check the following information is additionally signed: A time period *established-until* and information about a service to check the consent's actual status, named *status provider*. At the time of establishing consent, before signature generation, the data subject needs to encode the information for the status provider and optionally set the time limit. Thus, the data subject defines how the status provider can be contacted. Using time periods limits the time in which consented use can appear. If the time specified in *established-until* is reached, the consent is no longer established but vanished.

The status provider provides a function *check* to processors and third-parties. If the consent is still established, *check* yields a positive outcome. If *check* returns a negative result, this means that the consent has vanished. As a extension we envision to additionally provide information why the consent has vanished. The data subject controls the outcome of the *check* function. Using the status provider the processor queries the status of a given consent as it carries out the verify procedure. As only the data subject shall be allowed to give an authoritative answer, the outcome of *check* must be signed and contains a time-stamp. To retain a verifiable proof that, at the time of processing, the consent was still established, the processor saves the signed and time-stamped answer for his record.

This sounds familiar to the information and mechanisms usually found in public key certificates. In X.509 public key infrastructures (PKI) the revocation status of certificates is queried through certificate revocation lists (CRL) [7] or the online certificate status protocol (OCSP) [16]. A PKI can be used to determine the trust of the binding between the data subject's identifier and his public-key. Further, X.509 certificates and the revocation mechanisms can be facilitated to transport the information in our solution: The root hash value and tree information can be stored inside a X.509 certificate. Our previous work [18] showed the applicability for a similar use. Such a certificate will carry the status provider information, as a CRL distribution point. The time period will be stored as the certificate's validity period. The data subject's public-key, the root hash, and additional information can be stored as well. The certificate would be issued by the data subject, thus digitally signed using the data subject's private-key. So

all the relevant information is protected and it can be revoked without affecting the bond between the data subject and his key-pair.
Example: To participate in a raffle we consent that name and postal address are processed in order to send us potential winnings, but as we know the draw takes place in May, we could limit the established consent to the end of June. The vanished consent rendering all uses later than June "unconsented". Note aside, this could also be defined using privacy preferences that handle timing.

The data subject controls the information which status provider a processor uses to query the status of consent. Thus, logging revocation status checks allows the data subject to see for which complete signed data trees verification is requested. In other words, *check* is a call-back to the data subject, allowing him to see which of his data sets is in question of being processed. Allowing to only log on the level of data sets can be seen as a compromise, as it leaves some privacy for the data processor. Nevertheless both parties have a gain from using the status provider: The processor gains a re-assurance in the proof of the consent, and the data subject gains information about the re-use of his personal data.

4.4 Security in the Presence of an Attacker and Enforcement through Detection

We assume that strong asymmetric cryptography and cryptographically sound hash functions are used to generate the signature and the hash values. So an attacker without knowing the data subject's private-key can not modify or reproduce the expression of consent codified in the digital signature without knowing the appropriate values. However, there is no protection against "unconsented" data processing. Our approach deliberately chooses not to impose additional access control restrictions to the personal data. An attacker can simply ignore or change privacy preferences during processing, but he will not be able to present a proof of consent that yields a positive verification outcome. Thus, "unconsented" data processing destroys the verifiable proof of consent. With a system as ours in use, personal data without a verifiable consent has only little business value, because it can not be used further used to interact with the data subject or released to third-parties without the loss of consent going undetected. Thus, an attacker must fear legal implications or looses reputation when using or passing on unverifiable personal data.

5 Discussion

5.1 Related Work

Privacy compliant processing can either be guarded through access control and checked upfront or compliant processing can be checked and verified afterwards. Our work falls into the second category, but we see the two approaches as complimentary not as mutually exclusive. Examples of upfront checking approaches that can easily be augmented by our approach are work by Hutter et al [9] and Squicciarini et al. [19]. Hutter et. al. showed that while planning the composition and execution of compound web services the data owner's preferences with respect to privacy can be matched against each service's privacy policy, resulting in composition plans that use only web services that do not violate the given data's security policy. They use data flow analysis to see if the service plan respects the given privacy constraints and enforce this in a "trusted" composition engine. Squicciarini et. al. [19] defined in their work how data handling preferences set forth by the data subject can be matched against the policies of processing services. Their approach bases on policy subsumption. This allows to check whether the data can be handed over to the next service, because its handling preferences comply with the policies or not. Both offer no retainable proof of consent to benign parties involved.

In the case of checking privacy compliance afterwards the term of information accountability has been brought up by Weitzner et. al. [23]. They discuss the legal and technical framework to enable accountability for privacy. In another work Weitzner [22] showed how a simple logging facility, invoked whenever data is processed, can already help to detect privacy violations. Their work lacks a proof that processors, data subjects, and third-parties alike could verify.

Related in the field of aggregation is work done Devanbu et al. [3], Bertino et al. [4], and Carminati et al. [2]. Devanbu et al. use Merkle hash trees to verify the completeness of answers on queries. Their focus on verifying partial trees would be applicable for verification, but does not touch aggregation processes. The latter works [4] [2] protect data in transit without the need of trustworthy publishers, thus are more focussed on confidentiality.

We propose the use of digital signatures and suggest reusing existing PKI mechanisms. These mechanisms are well studied, as also are their overheads [15]. Our approach does not restrict the choice of the digital signature scheme. Schemes that need less resources for revocation could be used, for example Le et al. [11] proposed using a reverse chain of forward secure signature (FSS) in order invalidate certified credentials with minimal overhead from CA or OCSP/CRL involvement. Invalidating the private key s_x used for signing the credential $cred_x$ would also invalidate, due to the forward secrecy, all signed credentials with an index greater than x. This approach is not applicable in our case for two reasons: First, the data subject wants to revoke certain consent without affecting previously given consent. Second, the processor wants to retain a verifiable proof for actions in the past.

5.2 Performance

Obviously adding a signature adds an overhead over unsigned data, both in size and in performance. The increase in size is due to the need to additionally store and transmit the following information: a digital signature and an optional number of substitution hashes.

The data itself, and each data item, can be of arbitrary size, while the digital signature is of fixed length. Each substitution hash is of fixed length, regardless of the data item's or sub-tree's size it represents. When existing X.509 certificates are reused to store the digital signature, accompanied with validity periods and other relevant information, this would result in adding approximately 1200 bytes, including the begin and end markers. Even when using 2048 bit RSA modulus for the keys the certificates are usually smaller than 2000 bytes. Optional, if data-items are omitted, the needed substitution hashes (for example using SHA-1) occupy additional 160 bits. Of course using longer hash functions (SHA-256 or SHA-512) slightly increases this overhead. In general, the omission of a single data item does not reduce the data set's size by the size of the omitted data item, as the substitution hash needs to be stored. On the other hand, if several data items are to be omitted, and they form a sub-tree of the hash tree, they can be substituted by just one single hash value.

We will now look at the overhead in performance, and when and where it occurs. Generally speaking two steps involve processing time: The hash tree and the asymmetric cryptography. We shortly show how expensive each of these operations are, then we will look where and how often this additional computing is needed to generate, process, and check the verifiable proof of consent.

The hash tree we used in our prototype is a complete binary tree. The number of leaves equals the number of data items rounded up to the next even number. So the number of hash operations needed to generate the tree is $\approx (2 * \mathit{data\ items}) - 1$. Just to give you an indication, an Intel Core 2 Duo running at 2.4 GHz can roughly do 150 SHA-1 hashes on 1024 bits of data in 1 ms, and more on smaller data. Thus generating the root hash of a data set of 75 data items takes 1 ms. Since the original proposal by Merkle [14] in 1980, there have been several improvements on the generation and traversal of Merkle hash trees. In 2004 Szydlo [20] showed that it is possible to compute sequential tree leaves and authentication data in $2 \log_2(\mathit{leaves})$ time and $3 \log_2(\mathit{leaves})$ space.

Having the hash tree, the signature must be generated and verified. With RSA the signature generation is more labor intensive than the verification. The same Intel Core 2 Duo reported[4] about 30 signing and 1000 verify operations for 2048 bits per second. Having identified the performance overhead we will now look when and where it occurs.

The asymmetric distribution is helpful as the signature will be generated just once by the data subject during data dissemination. The data subject generates a single signature, once, when the consent is established. Additionally the data subject is not expected needing to submit high volumes of personal data in a short amount of time. On the other hand, the processors might need to process high volumes of personal data, but their operations are hash computations and signature verifications, both are less expensive.

Last, we look at the overhead required for querying the status provider. This results in communication overhead and one signature generation by the data subject or a service acting on his behalf and one signature verification by the processor that requested the status. Performance measures gathered from an unoptimized prototype that was implemented to secure the integrity and authenticity of fragmented web content [18] showed the following: Our Firefox

[4] Using `openssl speed` of OpenSSL 0.9.8h.

extension needed less than one second to parse a website's DOM tree, identify data-items, generate the hash tree, query the OCSP responder, and verify the certificate's digital signature, including the verification of the certificate chain.

We suppose the increase in size is marginally compared to todays bandwidth and communication costs. The increase in size can be limited if larger or multiple data items are omitted. Here the tree based structure optimally allows to replace several omitted data items by just one intermediate node's substitution hash. The cryptographic operations introduce another, more important, overhead, but as we showed the more expensive operations are distributed among the data subjects. The bulk processing done by processors only involves less costly operations of hashing and signature verification.

6 Conclusion

Our solution allows services to gain a proof of consent even for aggregated personal data. Doing so without the need of a direct relationship with the data subject and without the iterative involvement of the data subject. Additionally our solutions caters for changes in the expression of consent, allowing to vanish a once established consent. Technically, our solution builds on digitally signed hash tree and reuses PKI mechanisms, especially certificates and certificate revocation. As Weitzner et al. we see the desirable properties in accountability and allowing the data subject to follow the use of his personal data. We balanced the accountability with the reassurance gained through the consent status check. This results in a win-win situation, the processor is reassured of that the data subject's consent is still established and the data subject is able to see this as an entry in his accountability log.

Privacy protection based on access control, thus restricting the flow of personal data upfront, shall be used whenever possible. However, ongoing trends like copyleft licenses, advances in semantic web, and increased digital footprints through the social web [12] show the need for a digital form of the data subject's initial expression of consent. Our work offers a provable expression of consent, in a form that allows passing it on, that can be retained, and that remains verifiable by third-parties. This was not available for aggregated personal data before.

Augmenting existing access control based privacy protection architectures, we offer accountability based architectures a proof of consent that the social and legal frameworks can rely upon. The already existing services that process data can easily add the proposed mechanisms, as trust and revocation mechanisms can be borrowed from existing PKI or web of trust systems. New services and business models can be built upon the new proof of consent.

Further research will look into the adoption of other suitable digital signature mechanism, and better fit aggregated data. We plan to integrate our approach into existing web services.

References

1. Berners-Lee, T.: Semantic Web Road map (September 1998), http://www.w3.org/DesignIssues/Semantic.html
2. Carminati, B., Ferrari, E., Bertino, E.: Securing XML data in third-party distribution systems. In: Proceedings of 14th ACM CIKM, pp. 99–106 (2005)
3. Devanbu, P., Gertz, M., Kwong, A., Martel, C., Nuckolls, G., Stubblebine, S.: Flexible authentication of XML documents. In: 8th ACM Conf. on Computer and Comm. Security (2001)
4. Bertino, E., Carminati, B., Ferrari, E., Thuraisingham, B., Gupta, A.: Selective and authentic third-party distribution of XML documents. IEEE TKDE 16, 1263–1278 (2004)
5. EU. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data (October 1995)
6. Google. Google Friend Connect (May 2008), www.google.com/intl/en/press/annc/20080512_friend_connect.html
7. Housley, R., Polk, W., Ford, W., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280 (Proposed Standard), Updated by RFC 4325 (April 2000)
8. Housley, R., Polk, W., Ford, W., Solo, D.: RFC 3280 - internet X.509 PKI certificate and certificate revocation list (CRL) profile (April 2002)
9. Hutter, D., Volkamer, M.: Information flow control to secure dynamic web service composition. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 196–210. Springer, Heidelberg (2006)
10. Lai, Y.-L., Hui, K.L.: Internet opt-in and opt-out: investigating the roles of frames, defaults and privacy concerns. In: Shayo, C., Kaiser, K., Ryan, T. (eds.) CPR, pp. 253–263. ACM Press, New York (2006)
11. Le, Z., Ouyang, Y., Xu, Y., Ford, J., Makedon, F.: Preventing unofficial information propagation. In: ICICS, pp. 113–125 (2007)
12. Madden, M., Fox, S., Smith, A., Vitak, J.: PEW internet & american life project report: Digital footprints (December 2007), http://www.pewinternet.org/pdfs/PIP_Digital_Footprints.pdf
13. Merkle, R.C.: Secrecy, Authentication, and Public Key Systems, PhD thesis, Stanford (1979)
14. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy, p. 122 (1980)
15. Muñoz, J.L., Forné, J., Castro, J.C.: Evaluation of Certificate Revocation Policies: OCSP vs. Overissued-CRL. In: DEXA Workshops, pp. 511–518. IEEE Computer Society Press, Los Alamitos (2002)
16. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard) (June 1999)
17. O'Reilly, T.: What is Web 2.0 (September 2005), http://www.oreillynet.com/lpt/a/6228
18. Pöhls, H.C.: ConCert: Content revocation using certificates. In: Sicherheit 2008, Saarbrücken, Germany GI-Edition Lecture Notes in Informatics (LNI), vol. 128, pp. 149–162. GI (April 2008)

19. Squicciarini, A.C., Bhargav-Spantzel, A., Czeskis, A., Bertino, E.: Traceable and automatic compliance of privacy policies in federated digital identity management. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 78–98. Springer, Heidelberg (2006)
20. Szydlo, M.: Merkle tree traversal in log space and time. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027. Springer, Heidelberg (2004)
21. W3C. The platform for privacy preferences 1.0 (P3P1.0) specification (April 2002), http://www.w3.org/TR/P3P/
22. Weitzner, D.J.: Reciprocal Privacy (ReP) for the Social Web (December 2007), http://dig.csail.mit.edu/2007/12/rep.html
23. Weitzner, D.J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., Sussman, G.J.: Information accountability. Technical Report MIT-CSAIL-TR-2007-034, MIT (June 2007)

Embedding Renewable Cryptographic Keys into Continuous Noisy Data

Ileana Buhan, Jeroen Doumen, Pieter Hartel, Qiang Tang, and Raymond Veldhuis

Faculty of EWI, University of Twente, The Netherlands

Abstract. Fuzzy extractor is a powerful but theoretical tool to extract uniform strings from discrete noisy data. Before it can be used in practice, many concerns need to be addressed in advance, such as making the extracted strings renewable and dealing with continuous noisy data. We propose a primitive *fuzzy embedder* as a practical replacement for fuzzy extractor. Fuzzy embedder naturally supports renewability because it allows a randomly chosen string to be embedded. Fuzzy embedder takes continuous noisy data as input and its performance directly links to the property of the input data. We give a general construction for fuzzy embedder based on the technique of Quantization Index Modulation (QIM) and derive the performance result in relation to that of the underlying QIM. In addition, we show that quantization in 2-dimensional space is optimal from the perspective of the length of the embedded string. We also present a concrete construction for fuzzy embedder in 2-dimensional space and compare its performance with that obtained by the 4-square tiling method of Linnartz, *et al.* [13].

1 Introduction

Most cryptographic protocols rely on exactly reproducible key material. In fact, these protocols are designed to have a wildly different output if the key is perturbed slightly. Unfortunately, exactly reproducible keys are hard to come by, especially when they also need to have sufficient entropy. Luckily, it is relatively easy to find "fuzzy" sources, such as physically uncloneable functions (PUFs) [17] and biometrics [8]. However, such sources are inherently noisy and rarely uniformly distributed. The first (main) difficulty in transforming a fuzzy source into key material is to correct the noise and reproduce the same key every time. To solve this problem, the notion of secure sketch [12] has been proposed. The second difficulty lies in the fact the output of secure sketch may have a non-uniform distribution, while it should be as close to uniform as possible to serve as a cryptographic key. A strong randomness extractor could be used to turn the reproducible output into a nearly uniform string. In the literature, a common way of extracting keys from noisy data is to combine a secure sketch with a strong randomness extractor, which leads to the notion of a fuzzy extractor [8].

When deploying a fuzzy extractor in practice, more concerns need to be addressed. Firstly, even with the same input (noisy data), it should be possible to extract different keys (referred to as renewability). To achieve renewability, the (fixed) output of the fuzzy extractor must be randomized, for instance by using a common reference string. Unfortunately, this falls outside the scope of fuzzy extractor, even though it is

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 294–310, 2008.

recognized as an important and sensitive issue [2]. Secondly, fuzzy extractor only accepts discrete sources as input. Existing performance measures for secure sketches, such as entropy loss or min-entropy, lose their relevance when applied to continuous sources [12]. This limitation can be overcome by quantizing the continuous input. Li, *et al.* [12] propose to define relevant performance measures for secure sketch with respect to the chosen quantization method.

CONTRIBUTIONS. Our contribution is threefold. Firstly, we propose a new primitive *fuzzy embedder* which can be regarded as a practical replacement for fuzzy extractor. Fuzzy embedder can embed a uniformly distributed key while taking continuous noisy data as input. Its performance directly links to the property of the input data. Fuzzy embedder formalizes the concept of "key binding" in biometric template protection schemes surveyed by Uludag, *et al.* [20]. In fact, fuzzy embedder can also be regarded as a natural extension of fuzzy extractor, since it can embed a fixed string (for instance one obtained by applying a strong extractor to the input source) into a discrete source and thus achieve the same functionality, namely a randomized cryptographic key. However, a fuzzy embedder scheme can be directly used with any type of input to achieve the same goal as a fuzzy extractor scheme without the need to address those concerns mentioned previously.

Secondly, we propose a general construction for fuzzy embedder based on the technique of Quantization Index Modulation (QIM) and derive the performance result in relation to that of the underlying QIM. In the context of watermarking, using QIM can achieve efficient trade-offs between the information embedding rate, the reliability and the distortion [5]. The trade-offs of the underlying QIM give rise to similar trade-offs in fuzzy embedder performance measures. Note that shielding functions [13] can be regarded as a particular construction of a fuzzy embedder, as they focus on one particular type of quantizer. However, they only consider one-dimensional inputs.

Thirdly, we investigate different quantization strategies for high dimensional data and show that quantization in two dimensions gives an optimal length of the embedded uniform string. Finally, we propose a concrete construction of fuzzy embedder in 2-dimensional space and compare its performance with that obtained by the 4-square tiling method of Linnartz, *et al.* [13].

RELATED WORK. Dodis, *et al.* [8] consider discrete distributed noise and propose fuzzy extractors and secure sketches for different error models. These models are not directly applicable to continuously distributed sources. Linnartz, *et al.* [13] construct shielding functions for continuously distributed data and propose a practical construction which can be considered a 1-dimensional QIM. The same approach is taken by Li, *et al.* [12] who propose quantization functions for extending the scope of secure sketches to continuously distributed data. Buhan, *et al.* [3] analyze the achievable performance of such constructions given the quality of the source in terms of the false acceptance rate and false rejection rate of a biometric system.

The process of transforming a continuous distribution to a discrete distribution influences the performances of secure sketches and fuzzy extractors. Quantization is the process of replacing analogue samples with approximate values taken from a finite set of allowed values. The basic theory of one-dimensional quantization is reviewed by

Gersho [9]. The same author investigates the influence of high dimensional quantization on the performance of digital coding for analogue sources [10]. `QIM` constructions are used by Chen and Wornell [5] in the context of watermarking. The same authors introduce dithered quantizers [6]. Moulin and Koetter [16] give an excellent overview of `QIM` in the general context of data hiding. Barron, *et al.* [1] develop a geometric interpretation of conflicting requirements between information embedding and source coding with side information.

Fuzzy embedder is somehow related to the concept of information theoretic key agreement [14,15]. However, the settings of the problem are different. In secure message transmission based on correlated randomness the attacker and the legitimate participants have a noisy share of the same source data, while, in the fuzzy embedder setting, the attacker does not have access to the data source.

ROADMAP. The rest of the paper is organized as follows. In *Section* 2 we describe our notation and provide some background knowledge. In *Section* 3 we present the definition of fuzzy embedder and highlight the differences with fuzzy extractor. In *Section* 4 we propose a general construction of a fuzzy embedder from any `QIM` and express the performance in terms of the geometric properties of the underlying quantizers. In *Section* 5 we present a concrete construction for fuzzy embedder in 2-dimensional space and compare its performance with that obtained by the 4-square tiling method of Linnartz, *et al.*. In the last section we conclude this paper.

2 Preliminaries

Let $\mathcal{M}$ be an n-dimensional discrete, finite set, which together with a distance function $d_{\mathcal{M}} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+$ forms a metric space. Similarly, let $\mathcal{U}$ be an n-dimensional continuous domain, which together with the distance $d_{\mathcal{U}} : \mathcal{U} \times \mathcal{U} \to \mathbb{R}^+$ forms a metric space. For the purpose of this work, we use d for both $d_{\mathcal{M}}$ and $d_{\mathcal{U}}$. Capital letters are used to denote random variables while small letters are used to denote realizations of random variables. Continuous random variables are defined over the metric space $\mathcal{U}$ while discrete random variables are defined over the metric space $\mathcal{M}$. A random variable A is endowed with a probability density function $f_A(a)$. We use the random variable P when referring to public sketch data and R for random binary strings in the descriptions of fuzzy extractor and fuzzy embedder.

MUTUAL INFORMATION. By $I(A;B)$ we note the Shannon mutual information between the two random variables A and B, which measures the amount of uncertainty left about A when B is made public. We have $I(A;B) = 0$ if and only if A and B are independent random variables. Formal definitions of entropy, min-entropy, average min-entropy, and statistical distance SD can be found in [8].

FUZZY EXTRACTOR. According to the definition by Dodis, *et al.* [8], a fuzzy extractor extracts a uniformly random string r from a value x of random variable X in a noise-tolerant way with the help of some public sketch p (see, *Figure* 1). For a discrete metric space $\mathcal{M}$ with a distance measure d, fuzzy extractor [2,8] is formally defined as follows.

Definition 1 (Fuzzy Extractor). *An* $(\mathcal{M}, m, l, t, \epsilon)$ *fuzzy extractor is a pair of randomized procedures* $\langle \mathsf{Generate}, \mathsf{Reproduce} \rangle$ *with the following properties:*

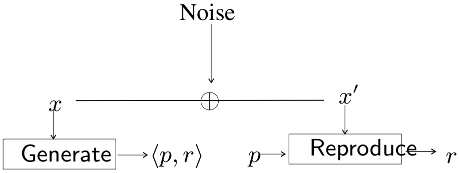


Fig. 1. A fuzzy extractor is a pair of two procedures $\langle \mathsf{Generate}, \mathsf{Reproduce} \rangle$. The Generate function takes noisy data x as input and returns a random string r and a public sketch p. The Reproduce function takes noisy data x' and the public sketch p as input, and outputs r if x and x' are close.

1. *The generation procedure on input of $x \in \mathcal{M}$ outputs an extracted string $r \in R = \{0,1\}^l$ and a public helper string $p \in P = \{0,1\}^*$.*
2. *The reproduction procedure takes an element $x' \in \mathcal{M}$ and the public string $p \in \{0,1\}^*$ as input. The* reliability *property of the fuzzy extractor guarantees that if $d(x, x') \leq t$ and r, p were generated by $(r, p) \leftarrow \mathsf{Generate}(x)$, then $\mathsf{Reproduce}(x', p) = r$. If $d(x, x') > t$, then no guarantee is provided about the output of the reproduction procedure.*
3. *The* security *property guarantees that for any random variable X with distribution $f_X(x)$ of min-entropy m, the string r is nearly uniform even for those who observe p: if $(r, p) \leftarrow \mathsf{Generate}(X)$, then $SD((R, P), (N, P)) \leq \epsilon$ where N is a random variable with uniform probability.*

In other words, a fuzzy extractor allows to generate the random string r from a value x. The reproduction procedure which uses the public string p produced by the generation procedure will output the string r as long as the measurement x' is close enough. This is the *reliability* property of the fuzzy extractor. The *security* property guarantees that r looks uniformly random to an attacker and her chance to guess its value from the first trial is approximately 2^{-m}. Security encompasses both *min-entropy* and uniformity of the random string r when p are known to an attacker.

We have two observations on the shortcomings of fuzzy extractor. One is that, the public string is from the discrete set $P = \{0,1\}^*$. However, there are biometric template protection schemes that fit the model of the fuzzy extractors for which P is drawn from $\mathbb{R}$ [13] or $\mathbb{Z}$ [18]. The other is that, defining min-entropy for X makes sense only if X has a discrete probability density function otherwise its min-entropy depends on the quantization of the variable [12].

QUANTIZATION. A continuous random variable A can be transformed into a discrete random variable by means of quantization, which we write as $Q(A)$. Formally, a quantizer is a function $Q : \mathcal{U} \rightarrow \mathcal{M}$ that maps $a \in \mathcal{U}$ into the closest *reconstruction point* in the set $\mathcal{M} = \{c_1, c_2, \cdots\}$ by

$$Q(a) = \mathsf{argmin}_{c_i \in \mathcal{M}} d(a, c_i)$$

where d is the distance measure defined on $\mathcal{U}$. The *Voronoi region* or the *decision region* of a reconstruction point c_i is the subset of all points in $\mathcal{U}$, which are closer to that particular reconstruction point than to any other reconstruction point. We denote

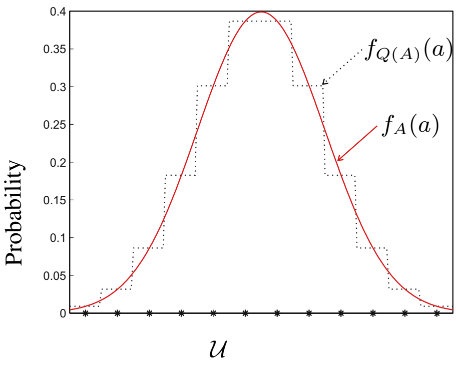


Fig. 2. By quantization, $f_A(a)$ (continuous line) is transformed into $f_{Q(A)}(a)$ (dotted line)

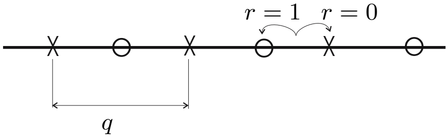


Fig. 3. Quantization of X with two scalar quantizers Q_0 and Q_1 both with step size q

with V_{c_i} the Voronoi region of reconstruction point c_i. When A is 1-dimensional, Q is called a *scalar* quantizer. If all Voronoi regions of a quantizer are equal, the quantizer is *uniform*. In the scalar case, the length of the Voronoi region is then called the *step size*. If the reconstruction points form a lattice, the Voronoi regions of all reconstruction points are congruent. By quantization, the probability density function of the continuous random variable A, $f_A(a)$ which is continuous, is transformed into the probability density function $f_{Q(A)}(a)$ which is discrete (See *Figure* 2).

QUANTIZATION-BASED DATA HIDING CODES. Quantization based data hiding codes, introduced by Chen, *et al.* [5] (also known as `QIM`), can embed secret information into a real value. We start with the following example.

Example 1. We want to embed one bit of information, thus $r \in \{0, 1\}$ into a real value x. For this purpose we use a scalar uniform quantizer with step size q, given by

$$Q(x) = q \left[\frac{x}{q} \right] .$$

The quantizer Q is used to generate a set of two new quantizers $\{Q_0, Q_1\}$ defined as:

$$v_0 = \frac{q}{4}, \;\; v_1 = -\frac{q}{4}, \;\; Q_0(x) = Q(x + v_0) - v_0, \;\; Q_1(x) = Q(x + v_1) - v_1 .$$

In *Figure* 3 the reconstruction points for the quantizer Q_1 are shown as circles and the reconstruction points for the quantizer Q_0 are shown as crosses. The embedding is done by mapping the point x to the elements of these two quantizers. For example, if $r = 1$, x is mapped to the closest $\circ$ point. The result of the embedding is the distance vector to the nearest $\times$ or $\circ$ as chosen by r. During reproduction procedure, when x is perturbed by noise, the quantizer will assign the received data to the closest $\times$ or $\circ$ point, and output 0 or 1 respectively.

Formally, a *Quantization Index Modulation* data hiding scheme, can be seen as `QIM` : $\mathcal{U} \times R \rightarrow \mathcal{M}$ a set of individual quantizers $\{Q_1, Q_2, \ldots Q_{2^l}\}$, where $l = |R|$ and each quantizer maps $x \in \mathcal{U}$ into a reconstruction point. The quantizer is chosen by the input value $r \in R$ such that $\mathtt{QIM}(x, r) = Q_r(x)$. The set of all reconstruction points is $\mathcal{M} = \bigcup_{r \in R} \mathcal{M}_r$ where $\mathcal{M}_r \subset \mathcal{M}$ is the set of reconstruction points of the quantizer Q_r.

We define the *minimum distance* $\sigma_{\min}$ of a QIM, as the minimum distance between reconstructions points of all quantizers in the QIM:

$$\sigma_{\min} = \min_{r_1, r_2 \in R} \min_{c^i_{r_1} \in \mathcal{M}_{r_1}, c^j_{r_2} \in \mathcal{M}_{r_2}} d(c^i_{r_1}, c^j_{r_2})$$

where $\mathcal{M}_{r_1} = \{c^1_{r_1}, c^2_{r_1}, \cdots\}$ and $\mathcal{M}_{r_2} = \{c^1_{r_2}, c^2_{r_2}, \cdots\}$. Hence, balls with radius $\frac{\sigma_{\min}}{2}$ and centers in $\mathcal{M}$ are disjoint. Let ζ_r be the smallest radius ball such that balls centered in the reconstruction point of quantizer Q_r with radius ζ_r cover the universe $\mathcal{U}$. We define the *covering distance* $\lambda_{\max}$ as:

$$\lambda_{\max} = \max_{r \in \mathcal{R}} \zeta_r.$$

Any ball $B(c, \zeta_r)$ contains at least one ball $B(c_r, \sigma_{\min}/2)$ for $c_r \in \mathcal{M}_r, \forall r \in R$. Hence, balls with radius $\lambda_{\max}$ and centers in $\mathcal{M}_r$ cover the universe $\mathcal{U}$.

A *dithered* QIM [6] is a special type of QIM for which all Voronoi region of all individual quantizers are congruent polytopes (generalization of a polygon to higher dimensions). Each quantizer in the ensemble $\{Q_1, Q_2, \ldots Q_{2^l}\}$ can be obtained by shifting the reconstruction points of any other quantizer in the ensemble. The shifts correspond to dither vectors $\{v_1, v_2, \ldots v_{2^l}\}$. The number of dither vectors is equal to the number of quantizers in the ensemble.

The reliability (or, the amount of tolerated noise) of a QIM is determined by the minimum distance between two neighboring reconstruction points. The size and shape (for high dimensional quantization) of the Voronoi region determines the tolerance for error. The number of quantizers in the QIM set determines the amount of information that can be embedded. By setting the number of quantizers and by choosing the shape and size of the decision region the performance properties can be fine tuned.

3 Fuzzy Embedder

In this section, we define fuzzy embedder and show its relationship with fuzzy extractor. It is worth stressing that the random key r is not extracted from the random x, but is generated independently, as illustrated in *Figure* 4.

Definition 2 (Fuzzy Embedder). *A* $(\mathcal{U}, \ell, \rho, \epsilon, \delta)$*-fuzzy embedder scheme consists of two polynomial-time algorithms* $\langle \mathsf{Embed}, \mathsf{Reproduce} \rangle$*, which are defined as follows:*

- Embed*:* $\mathcal{U} \times R \to P$*, where* $R = \{0, 1\}^l$*. This algorithm takes* $x \in \mathcal{U}$ *and* $r \in R$ *as input, and returns a public sketch* $p \in P$*.*
- $\mathsf{Reproduce}$*:* $\mathcal{U} \times P \to R$*. This algorithm takes* $x' \in \mathcal{U}$ *and* $p \in P$ *as input, and returns a string from* R *or an error symbol* $\perp$*.*

Given any random variable X *over* $\mathcal{U}$ *and a random variable* R*, the parameter* ρ, ϵ, δ *are defined as follows:*

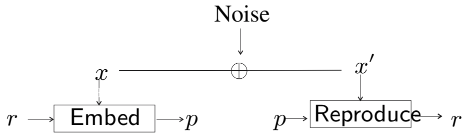


Fig. 4. A fuzzy embedder is a pair of two procedures $\langle \mathsf{Embed}, \mathsf{Reproduce} \rangle$. The Emded function takes noisy data x and a binary string r as input, and outputs a public sketch p. The Reproduce function takes noisy data x' and the public sketch p as input, and outputs r if x and x' are close.

- *The parameter ρ represents the probability that the fuzzy embedder can successfully reproduce the embedded key, and it is defined as*

  $$\rho = \min_{r \in R} \max_{x \in \mathcal{U}} \Pr(\mathsf{Reproduce}(x', \mathsf{Embed}(x, r)) = r | x' \in X).$$

  In the above definition, the maximum over $x \in \mathcal{U}$ ensures that we choose the best possible representative x for the random variable X. In most cases, this will be the mean of X.
- *The security parameter ϵ is equal to the mutual information between the embedded key and the public sketch, and it is defined as $\epsilon = I(R; \mathsf{Embed}(X, R))$.*
- *The security parameter δ is equal to the mutual information of the noisy data and the public sketch and is defined as $\delta = I(X; \mathsf{Embed}(X, R))$.*

Since the public sketch p is computed both on X and R, ϵ measures the amount of information revealed about X and δ measures the amount of information P reveals about the cryptographic key R. When evaluating security of algorithms, which derive secret information from noisy data, entropy measures like min-entropy, average min-entropy, and entropy loss are appealing since these measures have clear security applicability. However, these measures can only be applied to discrete random variable. In the case of continuous random variables, these measures depend on the precision used to represent the values of a random variable, as shown in the following example.

Example. Assume that all points X are real numbers between $[0, 1]$ and are uniformly distributed. Assume further that points in X are represented with 2-digit precision, which leads to a min-entropy $H_\infty(X) = \log_2 100$. If we choose to represent points with 4-digit precision the min-entropy of X becomes $H_\infty(X) = \log_2 10000$, which is higher then $H_\infty(X) = \log_2 100$ although in both cases X is uniformly distributed over the interval $[0, 1]$.

More examples related to average min-entropy and entropy loss can be found in the work of Li *et al.* [12]. We have chosen mutual information because it captures the measure of dependence between two random variables regardless of their types of distributions (discrete or continuous).

FUZZY EXTRACTOR AND FUZZY EMBEDDER. From *Definitions* 1 and 2, we argue that a fuzzy embedder may be more appealing than fuzzy extractor in practice, due to the following reasons:

1. A fuzzy embedder scheme accepts continuous data as input and can embed different keys. In contrast, in a practical deployment, a fuzzy extractor scheme must be

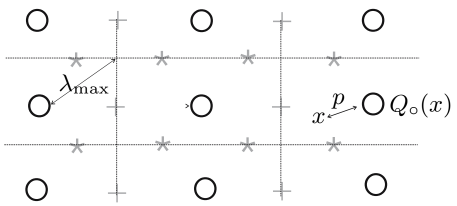


Fig. 5. Embed function of *QIM*-fuzzy embedder

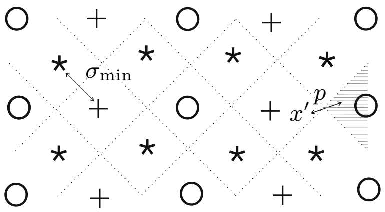


Fig. 6. Reproduce function of a *QIM*-fuzzy embedder

combined with quantization and re-randomization to achieve the same goals as a fuzzy embedder.

2. A fuzzy embedder construction leads to a fuzzy extractor construction. Given a $(\mathcal{U}, \ell, \rho, \epsilon, \delta)$-fuzzy embedder scheme, we can construct a fuzzy extractor scheme $\langle \mathsf{Generate}', \mathsf{Reproduce}' \rangle$ as follows:
   - $\mathsf{Generate}'$: $\mathcal{U} \to P \times R$. This algorithm takes $x \in \mathcal{U}$ as input, chooses $r \in R$, and returns $p = \mathsf{Embed}(x, r)$ and r.
   - $\mathsf{Reproduce}'$: $\mathcal{U} \times P \to R$. This algorithm takes $x' \in \mathcal{U}$ and $p \in P$ as input, and returns the value $\mathsf{Reproduce}(x, p)$.

4 A Practical Construction for Fuzzy Embedder

In this section, we present a general construction for fuzzy embedder using a QIM and analyze the performance of this construction in terms of reliability and security. We also investigate optimization issues when $\mathcal{U}$ is n-dimensional.

QIM-FUZZY EMBEDDER. A fuzzy embedder can be constructed from *any* QIM by defining the embed procedure as:

$$\mathsf{Embed}(x, r) = \mathtt{QIM}(x, r) - x,$$

and the reproduction procedure as the minimum distance Euclidean decoder:

$$\mathsf{Reproduce}(x', p) = \widetilde{Q}(x' + p),$$

where $\widetilde{Q} : \mathcal{U} \to R$ is defined as

$$\widetilde{Q}(y) = \operatorname*{argmin}_{r \in R} d(y, \mathcal{M}_r).$$

Intuitively, our construction is a generalization of the scheme of Linnartz, *et al.* [13]. *Figures* 5 and 6 illustrate Embed and Reproduce, respectively, for a QIM ensemble of three quantizers $\{Q_\circ, Q_+, Q_\star\}$. During embedding, the secret $r \in \{\circ, \star, +\}$ selects a quantizer, say $Q_\circ$. The selected quantizer finds the reconstruction point $Q_\circ(x)$ closest to x and the embedder returns the difference between the two as p, with $p \leq \lambda_{\max}$. Reproduction from p and x' should return $\circ$ only if $x' + p$ is in one of the Voronoi regions of $Q_\circ$ (hatched area in *Figure* 6). Errors occur if $(x' + p)$ is not in any of the Voronoi regions of $Q_\circ$, thus the size and shape (for $n \geq 2$) of the Voronoi region param eterized by the radius of the inscribed ball $\sigma_{\min}/2$ determines the probability of errors.

RELIABILITY. In the following lemma, we link the reliability of a QIM-fuzzy embedder to the size and shape of the Voronoi regions of the employed QIM.

Lemma 1 (Reliability). *Let $\langle \mathsf{Embed}, \mathsf{Reproduce}\rangle$ be a $(\mathcal{U}, \ell, \rho, \epsilon, \delta)$ QIM-fuzzy embedder, and let X be a random variable over $\mathcal{U}$ with joint density function $f_X(x)$. For any $r \in R$, we define*

$$\rho(r) = \int_{\mathcal{V}_r} f_X(y - \mathsf{Embed}(X, r))dy,$$

where $\mathcal{V}_r = \bigcup_{c \in \mathcal{M}_r} V_c$ is the union of the Voronoi regions of all reconstruction points in $\mathcal{M}_r$. Then the reliability is equal to

$$\rho = \min_{r \in R} \rho(r).$$

Proof: Since $\rho(r)$ is exactly the probability that an embedded key r will be reconstructed correctly, the statement follows from the definition. □

Most known noisy data, such as biometrics and PUFs, have two main properties: larger distances between x and the measurement x' are increasingly unlikely, and the noise is not directional. Thus the primary consideration for reliability is the size of the inscribed ball of the Voronoi regions, which has radius $\sigma_{\min}/2$.

Corrolary 1 (Bounding ρ). *In the settings of Lemma 1, the reliability parameter ρ can be bounded by*

$$\min_{r \in R} \sum_{c \in \mathcal{M}_r} \int_{B(c, \frac{\sigma_{\min}}{2})} f_X(y)dy \leq \rho$$

where $B(c, r)$ is the ball centered in c with radius r.

Proof. The above relation follows from the definition of reliability, since $S(c, \frac{\sigma}{2}) \subset V_c$ and $x + \mathsf{Embed}(X, r)$ is always a reconstruction point. □

Corollary 1 shows that reliability is at least the sum of all balls of radius $\frac{\sigma_{\min}}{2}$ inscribed in the Voronoi regions. Thus the size of the inscribed ball is an important parameter, which determines the reliability to noise.

SECURITY. In our construction, if an attacker learns the value x she can reproduce the value r from p. However, if it learns the secret key r, she could cannot exactly reproduce x, which is further illustrated in the following example

Example. In the fuzzy embedder example given in *Figure* 6, the attacker can choose between three different key values$\{\circ, +, \star\}$. Assume she learns the correct key, in our example $\circ$. To find the correct value for x she still has to decide which of the reconstruction points of the quantizer $Q_\circ$ is closest to x. Without any other information this is an impossible task since the quantizer $Q_\circ$ has an infinite number of reconstruction points.

Since the full disclosure of the string r is not enough to recover x, we can conclude that $\epsilon \leq \delta$. We now consider how large δ, the leakage on the key depending on P, which is a continuous variable in our construction. We know that any $p \in P$ has the property that $p \leq \lambda_{\max}$. A technical difficulty in characterizing the size of P arises as P is not necessarily discrete. Tuyls, *et al.* [19] show the following result, establishing a link between the continuous and the quantized version of P denoted here with P_d.

Lemma 2 (Tuyls et al. [19]). *For continuous random variables X, Y and $\xi > 0$, there exists a sequence of discretized random variables X_d, Y_d that converge pointwise to X, Y (when $d \to \infty$) such that for sufficiently large d, $I(X;Y) \geq I(X_d;Y_d) \geq I(X;Y) - \xi$.*

Since $I(R;P_d) \leq H(P_d) \leq |P_d|$, where $|P_d|$ is the size of the sketch. Thus it is best to have $|P_d|$ as small as possible. In our construction, we have $|P_d| \leq \lambda_{\max}$. Thus by bounding the size of p we bound the value of δ.

OPTIMIZATION. In this paragraph, we analyze the key length allowed by the restrictions placed by our performance criteria on the embed and reproduce procedures. Firstly, we take a look at the reproduce procedure which ties directly with the reliability. The minimum size of an error to produce a wrong decoding is $\sigma_{\min}/2$. Thus, the collection of balls centered in the reconstruction point of all quantizers with radius $\sigma_{\min}/2$ should be disjoint.

Secondly, the embed procedure has to be able to embed any key $r \in R$ into an arbitrary point x. Hence, for each key r the collection of balls centered in the reconstruction

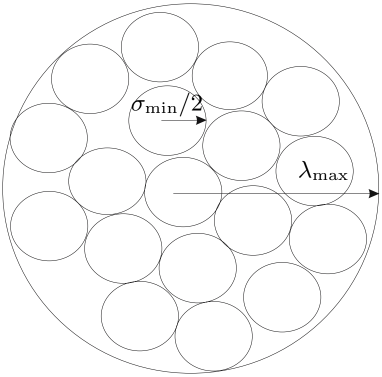


Fig. 7. *Optimization of reliability versus security. Reliability is determined by the size of the ball with radius $\sigma_{\min}/2$. Each small ball has associated to its center a different key $r \in R$. The number of small ball inside the large ball with radius $\lambda_{\max}$ is at least 2^l the number of elements in R. To have as many keys as possible we want to increase the number of small ball, thus we want* dense (sphere) packing. *The size of the public sketch $p \in P$ is at most $\lambda_{\max}$. Since for any $x \in \mathcal{U}$ we want to be within λ_{max} distance to a specific $r \in R$, large balls should* cover optimally *the space $\mathcal{U}$. When the point x falls in a region, which does not belong to any ball the reproduction procedure gives the closest center of a small ball, thus* we want polytopes which tile the space.

points of Q_k and with radius $\lambda_{\max}$ should cover the entire space $\mathcal{U}$. $\lambda_{\max}$ and $\lambda_{\min}$ can be linked as follows:

Lemma 3. *The covering distance of a* `QIM`*, defined in Section 2, is bounded by:*

$$\lambda_{\max} \geq \sqrt[n]{N}\frac{\sigma_{\min}}{2}$$

where n represents the dimension of the universe $\mathcal{U}$ and N is the number of different quantizers.

Proof: As noted above, all balls with radius $\sigma_{\min}/2$ centered in the centroids of the whole ensemble are disjoint. Each collection of balls with radius $\lambda_{\max}$ centered in the centroids of an individual quantizer gives a covering of the space $\mathcal{U}$, see *Figure* 7. Therefore, a ball with radius $\lambda_{\max}$, regardless of its center, contains at least the volume of N disjoint balls of radius $\sigma_{\min}/2$, one for each quantizer in the ensemble. Comparing the volumes, we have

$$s_n \lambda_{\max}^n \geq s_n N (\frac{\sigma_{\min}}{2})^n$$

where s_n is a constant only depending on the dimension. □

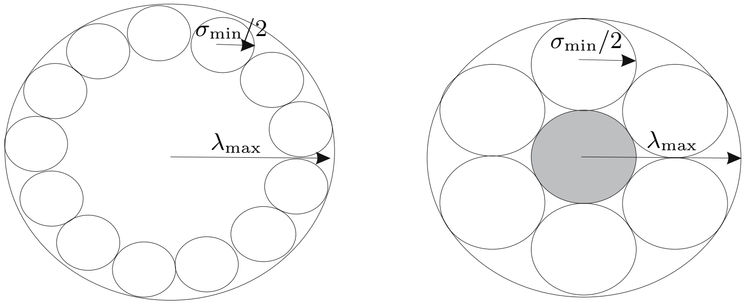
Consider the case when an intruder has partial knowledge about the random variable X. For example, she could know the average distribution of all (fingerprint) biometrics, or the average distribution of the PUFs. This average distribution is known in the literature as background distribution. While any `QIM`-fuzzy embedder achieves equiprobable keys if the background distribution on $\mathcal{U}$ is uniform, the equiprobability can break down when this background distribution is non-uniform and known to the intruder. A legitimate question is: *how can a* `QIM`*-fuzzy embedder achieve equiprobable keys when the background distribution is not uniform?*

In the literature [4,7,13], it is often assumed that the background distribution is a multivariate Gaussian. We make a much weaker assumption, namely the background distribution is not uniform but spherically symmetrical and decreasing. In other words, we assume that measurement errors of the noisy data only depend on the distance, and not on the direction, and that larger errors are less likely.

Thus, to achieve equiprobable keys given this background distribution, the reconstruction points must be equidistant as for example the construction in *Figure* 8 (a). Note that putting more small balls inside the large ball is not possible since they are not equiprobable. The problem with the construction in *Figure* 8 (a) is the size of the sketch which becomes large.

The natural question, which arise is: *what is the minimum sketch size attainable such that all keys are equiprobable for a given desired reliability?* This question naturally leads us to consider the kissing number $\tau(n)$, which is defined to be the maximum number of white n-dimensional spheres touching a black sphere of equal radius, see *Figure* 8 (b). The radius of the small balls determines reliability and the minimum $\lambda_{\max}$, such that a `QIM`-fuzzy embedder can be built is equal to the radius of the circumscribed ball of as shown in *Figure* 8 (b).

The next question we ask is: *for a minimum sketch size and a given reliability, are there dimensions which are better then others?* For example why not pack spheres in



(a) (b)

Fig. 8. (a) Construction which yields equiprobable keys in case the background distribution is spherical symmetrical in the two dimensional space. (b) Optimal construction which results in minimal public sketch size and has equiprobable keys in the two dimensional space.

the three dimensional space where the kissing number is 12. For the same reliability it is possible to obtain more keys? For most dimensions, only bounds on the kissing number are known [11,21]. Assuming a spherically symmetrical and decreasing background distribution, we have the following bound on equiprobable keys.

Theorem 1 (Optimal high dimensional packing.). *Assume the background distribution to be spherically symmetrical and decreasing. For a* $(\mathcal{U}, \ell, \rho, \epsilon, \delta)$ `QIM`*-fuzzy embedder with* $\dim(\mathcal{U}) = n$ *with equiprobable keys and minimal sketch size, we have that* $\ell \leq \tau(n)$.

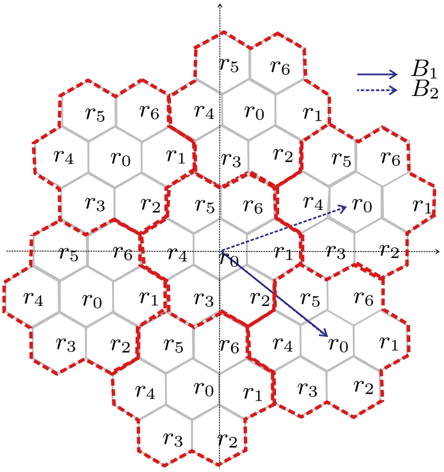
Proof sketch: The target reliability ρ_0 will translate to a certain radius σ_0. In other words, we need to stack balls of radius σ_0 optimally. To achieve the maximum number of equiprobable keys without the sketch size getting too big, the best construction is to center the background distribution in one such ball, and to assign a different key to each touching ball. Thus the amount of possible equiprobable keys is upper bounded by the kissing number $\tau(n)$. □

From the known bounds on the kissing number [11,21], we have the following somewhat surprising conclusion:

Corrolary 2. *Assuming a spherically symmetrical and decreasing background distribution on* $\mathcal{U}$ *and equiprobable keys, for a* $(\mathcal{U}, \ell, \rho, \epsilon, \delta)$ `QIM`*-fuzzy embedder the most equiprobable keys are attained by quantizing two dimensions at a time, leading to* $N(n)$ *different keys, where*

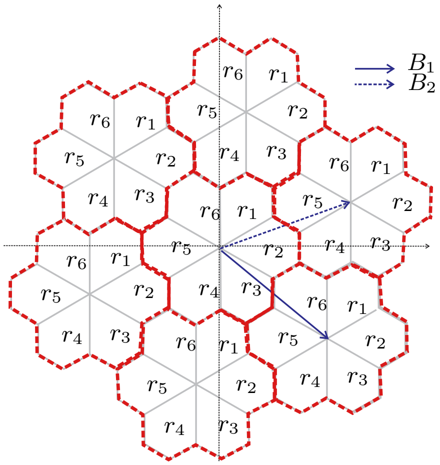
$$N(n) = 6^{\lfloor \frac{n}{2} \rfloor} 2^{(n - 2\lfloor \frac{n}{2} \rfloor)}.$$

Proof: Known upper bounds [11] on the kissing number in n dimensions state that $\tau(n) \leq 2^{0.401n(1+o(1))}$. This means that $N(n) \geq \tau(n)$ in all dimensions, since $N(n) \approx 2^{1.3n}$ and small dimensions can easily be verified by hand. Also note that $N(n_1 + n_2) \leq N(n_1)N(n_2)$. Thus quantizing dimensions pairwise gives the largest number of equiprobable keys for any spherically symmetric distribution. □



r_5 r_6
r_5 r_6 r_4 r_0 r_1
r_4 r_0 r_1 r_3 r_2 r_5 r_6
r_3 r_2 r_5 r_6 r_4 r_0 r_1
r_5 r_6 r_4 r_0 r_1 r_3 r_2
r_4 r_0 r_1 r_3 r_2 r_5 r_6
r_3 r_2 r_5 r_6 r_4 r_0 r_1
r_4 r_0 r_1 r_3 r_2
r_3 r_2
B_1
B_2

Fig. 9. Reproduce function of 7-hexagonal tiling



r_6 r_1
r_5 r_2
r_6 r_1
r_5 r_2 r_4 r_3 r_6 r_1
r_4 r_3 r_6 r_1 r_5 r_2
r_6 r_1 r_5 r_2 r_4 r_3
r_5 r_2 r_4 r_3 r_6 r_1
r_4 r_3 r_6 r_1 r_5 r_2
r_5 r_2 r_4 r_3
r_4 r_3
B_1
B_2

Fig. 10. Reproduce function of 6-hexagonal tiling

5 QIM-Fuzzy Embedder from 2-Dimensional Quantization

In this section we present our main construction, referred to as 6-hexagonal tiling, of QIM-fuzzy embedder by quantizing 2-dimensional subspaces of continuous and noisy data. We compare the performance with the 4-square tiling method introduced by Linnartz, *et al.* [13].

Preliminary concept. Let the continuous and noisy data be represented with a n-dimensional variable $X = (X_1, X_2, \cdots X_n)$. We assume that n is even; otherwise one of the vector elements can be quantized with a 1-dimensional QIM as the one in our example in *Section* 2. Thus, X can be partitioned into $\frac{n}{2}$ 2-dimensional subspaces and each one can be considered separately. We take the subspace (X_1, X_2) as an example in the rest of this section. On the x-axis in *Figure* 9 we have the values for X_1 and on the y-axis we have the values of X_2. Along the z-axis (not shown in the figure) we have the joint probability density $f_{X_1X_2}(x)$.

Naturally, we want to choose the densest circle packing for the 2-dimensional space, where all circles have equal radius and the center of the circle is the reconstruction point which is associated with a key value. However, the circles do not tile the space so that, when x (the realization of X) falls into the non-covered region it cannot be associated with any reconstruction point. Therefore, we need to approximate the circle with some polygons that can tile the space. In 2-dimensional space, there are only three types of polygons: triangle, square, and hexagon. Since we assume a spherical symmetrical distribution for $f_{X_1X_2}$, hexagon is the best approximation to the circle from the reliability point of view.

5.1 Description of 6-Hexagonal Tiling

First attempt. In our construction, the reconstruction points of all quantizers are shifted versions of some base quantizer Q_0. A dither vector $\overrightarrow{v_r}$ is defined for each possible $r \in \mathcal{R}$. We define the *tiling polygon* as the repeated structure in the space that is obtained by decoding to the closest reconstruction point. It follows from this definition that the *tiling polygon* contains exactly one Voronoi region for each quantizer in the ensemble. In *Figures* 9 the *tiling polygons* are delimited by the dotted line. More specifically, we define a dithered QIM using an ensemble of 7 quantizers. The reconstruction points of the base quantizer Q_0 are defined by the lattice spanned by the vectors $\overrightarrow{B_1} = (5, \sqrt{3})q$, $\overrightarrow{B_2} = (4, -2\sqrt{3})q$, where q is the scaling factor of the lattice. In *Figure* 9 these points are labeled r_0. The other reconstruction points of quantizers Q_i $(1 \leq i \leq 6)$ are obtained by shifting the base quantizer by the dither vectors $\{\overrightarrow{v_1}, \cdots, \overrightarrow{v_6}\}$ such that $Q_i(x) = Q_0(\overrightarrow{x} + \overrightarrow{v_i})$. The values for these dither vectors are: $\overrightarrow{v_1} = (2, 0)$, $\overrightarrow{v_2} = (-3, \sqrt{3})$, $\overrightarrow{v_3} = (-1, -\sqrt{3})$, $\overrightarrow{v_4} = (-2, 0)$, $\overrightarrow{v_5} = (3, -\sqrt{3})$, and $\overrightarrow{v_6} = (1, \sqrt{3})$. The embed and reproduce procedures are defined in *Section* 4.

This construction (referred to as 7-hexagonal tiling) can embed $n \times \frac{\log_2 7}{2}$ bits, where n is the dimensionality of random variable X. It is optimal from the reliability point of view. However, assume that the background distribution is a spherical symmetrical distribution with mean centered in the origin of the coordinates. In the construction above the hexagon centered in the origin will typically have a higher associated probability than the off-center hexagons. This effect grows as we increase the scaling factor q of the lattice. Therefore, keys might be not equiprobable when the background distribution is not flat enough.

Improved construction. In the improved construction, namely 6-hexagonal tiling, we eliminate the middle hexagon to make all keys equiprobable (see *Figure* 10). Consequently, the tiling polygon is formed by 6 decision regions and thus there are only 6 dither vectors. As a result, the dither vectors, $\{\overrightarrow{v_1}, \cdots, \overrightarrow{v_6}\}$ are used to construct the quantizers, but the basic quantizer Q_0 itself is not used. The embed and reproduce procedures remain the same.

Our main construction can embed $n \times \frac{\log_2 6}{2}$ bits, where n is the dimensionality of random variable X. Compare with the first attempt, this construction is not optimal from the key length point of view. However, keys are equiprobable regardless of the background distribution, which we regard to be more favorable in cryptographic applications.

5.2 Comparison with 4-Square Tiling

We compare the performance between 6-hexagonal tiling and 4-square tiling in terms of reliability, the key length, and mutual information. Here we consider identically and independently distributed (i.i.d) Gaussian sources. We assume that the background distribution has mean $(0, 0)$ and standard deviation ${\sigma_{X_1 X_2}}^2$. We also assume that for any random $(X_1, X_2) \in \mathcal{U}^2$, the probability distribution of $f_{X_1 X_2}(x)$ has mean $\mu = (\mu_1, \mu_2)$ and standard deviation σ_x^2. Note that these assumptions are abstracted from the area of biometrics (as an example of continuous and noisy data).

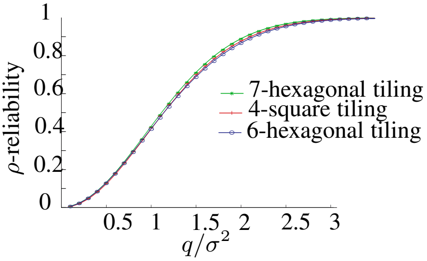
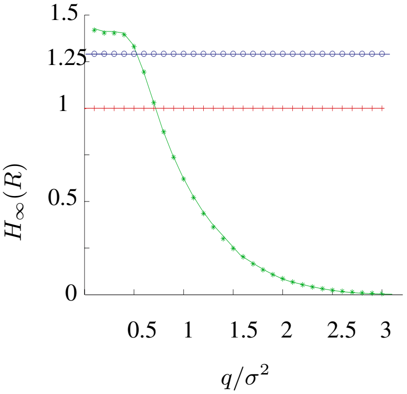


Fig. 11. Reliability of the three $\mathtt{QIM}$-fuzzy embedder constructions



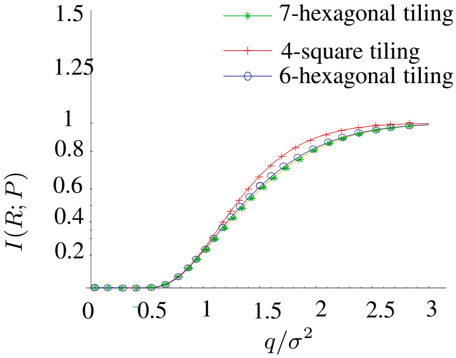


Fig. 12. Key length comparison for the three QIM-fuzzy embedder constructions-scaled to one dimension

Fig. 13. Mutual information between the key and the public sketch for the three $\mathtt{QIM}$-fuzzy embedders

To evaluate the reliability relative to the quality of the source data (i.e., the amount of noise measured in the terms of standard deviation from mean), we compute probabilities associated with equal area decision regions, and the reconstruction point centered in the mean μ of the distribution $f_X(x)$. The curves in Figure 11 were obtained by progressively increasing the area of the Voronoi regions. The size of Voronoi region is controlled by the scaling factor of the lattice, namely q. From the figure, our 6-hexagonal tiling construction has a slightly better performance than the 4-square tiling method. This is because the regular hexagon best approximates a circle, the optimal geometrical form for a spherical symmetrical distribution. The key-length comparison is shown in Figure 12. Clearly, our 6-hexagonal tiling construction has a significantly better performance than the 4-square tiling method. Note that maximizing the key length means minimizing the probability for an attacker to guess the key correctly on her first try. The comparison of mutual information for the key when publishing the sketch is shown in Figure 13. Note that the values are scaled to the number of bits lost from each bit that is made public. From the figure, our 6-hexagonal tiling construction has a slightly better performance than the 4-square tiling method.

6 Conclusion

We have proposed a new primitive *fuzzy embedder* as a practical replacement for fuzzy extractor. Fuzzy embedder has solved two practical problems encountered when a fuzzy extractor scheme is used in practice: (1) fuzzy embedder naturally supports renewability, and (2) it supports direct analysis of quantization effects. We have also proposed a general construction of fuzzy embedder using a QIM. The QIM performance measures (in the context of watermarking) can be directly translated into the reliability and security properties of the constructed fuzzy embedder. When considering equiprobable keys, we have shown that quantizing dimensions pairwise gives the largest key length. We have proposed a concrete construction, namely 6-hexagonal tiling, and shown that it has a better performance than the 4-square tiling method introduced by Linnartz, *et al.* [13].

References

1. Barron, R.J., Chen, B., Wornell, G.W.: The duality between information embedding and source coding with side information and some applications. IEEE Transactions on Information Theory 49(5), 1159–1180 (2003)
2. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 82–91. ACM, New York (2004)
3. Buhan, I., Doumen, J., Hartel, P.H., Veldhuis, R.N.J.: Fuzzy extractors for continuous distributions. In: Deng, R., Samarati, P. (eds.) Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS), pp. 353–355. ACM, New York (2007)
4. Chang, Y.J., Zhang, W., Chen, T.: Biometrics-based cryptographic key generation. In: International Conference on Multimedia and Expo (ICME), pp. 2203–2206. IEEE, Los Alamitos (2004)
5. Chen, B., Wornell, G.W.: Quantization Index Modulation Methods for Digital Watermarking and Information Embedding of Multimedia. The Journal of VLSI Signal Processing 27(1), 7–33 (2001)
6. Chen, B., Wornell, G.W.: Dither modulation: a new approach to digital watermarking and information embedding. In: Proceedings of SPIE, vol. 3657, p. 342 (2003)
7. Chen, C., Veldhuis, R.N.J., Kevenaar, T.A.M., Akkermans, A.H.M.: Multi-bits biometric string generation based on the likelyhood ratio. In: IEEE conference on Biometrics: Theory, Applications and Systems, pp. 1–6. IEEE, Los Alamitos (2007)
8. 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)
9. Gersho, A.: Principles of quantization. IEEE Transactions on Circuits and Systems 25(7), 427–436 (1978)
10. Gersho, A.: Asymptotically optimal block quantization. IEEE Transactions on Information Theory 25(4), 373–380 (1979)
11. Kabatiansky, G.A., Levenshtein, V.I.: Bounds for packings on a sphere and in space. Problemy Peredachi Informatsii 1, 3–25 (1978)
12. Li, Q., Sutcu, Y., Memon, N.: Secure sketch for biometric templates. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 99–113. Springer, Heidelberg (2006)

13. Linnartz, J.P., Tuyls, P.: New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 393–402. Springer, Heidelberg (2003)
14. Maurer, U.: Perfect cryptographic security from partially independent channels. In: Proceedings of the 23rd ACM Symposium on Theory of Computing (STOC), pp. 561–572. ACM Press, New York (1991)
15. Maurer, U.: Secret key agreement by public discussion. IEEE Transaction on Information Theory 39(3), 733–742 (1993)
16. Moulin, P., Koetter, R.: Data-hiding codes. Proceedings of the IEEE 93(12), 2083–2126 (2005)
17. Skoric, B., Tuyls, P., Ophey, W.: Robust key extraction from physical uncloneable functions. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 407–422. Springer, Heidelberg (2005)
18. Tuyls, P., Akkermans, A., Kevenaar, T., Schrijen, G., Bazen, A., Veldhuis, R.: Practical biometric authentication with template protection. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 436–446. Springer, Heidelberg (2005)
19. Tuyls, P., Goseling, J.: Capacity and examples of template-protecting biometric authentication systems. In: Maltoni, D., Jain, A.K. (eds.) BioAW 2004. LNCS, vol. 3087, pp. 158–170. Springer, Heidelberg (2004)
20. Uludag, U., Pankanti, S., Prabhakar, S., Jain, A.K.: Biometric cryptosystems: Issues and challenges. Proceedings of the IEEE 92(6), 948–960 (2004)
21. Zeger, K., Gersho, A.: Number of nearest neighbors in a euclidean code. IEEE Transactions on Information Theory 40(5), 1647–1649 (1994)

Automated Device Pairing for Asymmetric Pairing Scenarios

Nitesh Saxena and Md. Borhan Uddin

Computer and Information Science
Polytechnic Institute of New York University
Brooklyn, NY 11201, USA
nsaxena@poly.edu, borhan@cis.poly.edu

Abstract. "Secure Device Pairing" is the process of bootstrapping secure communication between two human-operated devices over a short- or medium-range wireless channel (such as Bluetooth, WiFi). The devices in such a scenario can neither be assumed to have a prior context with each other nor do they share a common trusted 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.

Recently proposed pairing protocols are based upon bidirectional physical channels. However, various pairing scenarios are asymmetric in nature, i.e., only a unidirectional physical channel exists between two devices (such as between a cell phone and an access point). In this paper, we concentrate on pairing devices using a unidirectional physical channel and analyze recently proposed protocol on this topic [14]. Moreover, as an improvement to [14], we present an efficient implementation of a unidirectional physical channel based on multiple blinking LEDs as transmitter and a video camera as a receiver.

Keywords: Distributed Protocols, Mobile/Ad-Hoc Systems, Security.

1 Introduction

Short-range wireless communication, based on technologies such as Bluetooth and WiFi, is becoming increasingly popular and promises to remain so in the future. With this surge in popularity, come 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, and so on. Pairing would be easy to achieve, if there existed a global infrastructure 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 real-world research problem. The problem has been at the forefront of various recent standardization activities, see [20].

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 311–327, 2008.

A recent research direction to pairing is to use an auxiliary physically authenticatable channel i.e., physical channel, also 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 channels, etc. Unlike the wireless channel, on the OOB channel, an adversary is assumed to be incapable of modifying messages, however, it can eavesdrop on, 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 requires at least 80 to 160 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, [7] and [9], reduce the length of data to be transmitted over the OOB channels to only 15 bits or so. The concept of SAS-based authentication was first introduced by Vaudenay in [22].

Based on the above-mentioned protocols, a number of pairing schemes with various OOB channels have been proposed. We review these in the next section. In this paper, we concentrate on pairing devices using *unidirectional* OOB channels. The motivation for this is that in various pairing scenarios, bidirectional d2d channels do not exist because only one of the devices being paired has a receiver (such as while pairing wifi laptop and a cell phone). Since receivers are generally expensive, it is not feasible to add them onto commodity devices, such as access points, bluetooth headsets, etc. Moreover, even in scenarios, where bidirectional d2d or the equivalent bidirectional d2h-h2d channels do exist, it is always beneficial to use only one of them for efficiency and usability reasons.

With the above motivation, we take a closer look at our previously proposed protocol that can be used for pairing two devices using a “short” unidirectional OOB channel in one direction and a unidirectional “single-bit” OOB channel in the other direction [14]. Since a “single-bit” channel is easy and fast to implement, we ignore this bidirectionality and from here on, refer to the protocol of [14] as a protocol that can pair two devices using a unidirectional OOB channel. The protocol is reviewed in next section.

Our Contributions. In this paper, we make twofold contributions:

- First, we analyze the protocol of [14] (as it did not come with a security proof). We show that the protocol is insecure in a security model that allows an adversary to delay/replay information transmitted over the OOB channels. In fact, we argue that in such a model, it is impossible to achieve pairing with only a unidirectional OOB channel. Next, we consider a weaker yet practical security model that does not allow an adversary to delay/replay messages over the OOB channel and prove that the protocol of [14] indeed remains secure in this model.
- Second, as an improvement to [14], we propose a new implementation of a OOB channel using LEDs as transmitter and video camera as receiver. Unlike the results of [14], the implementation of our channel is much more efficient and its bandwidth improves with the increase in the number of LEDs. Since most devices have multiple LEDs (and if not, they can be cheaply added on), our implementation is an efficient way to pair two devices (such as headset and camera phone, access point and camera phone), one of which has a video camera. Our implementation has other useful applications in Bluetooth/WiFi device discovery, sensor network key distribution and in general, in data transmission.

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 analyze the protocol of [14]. Finally, in Section 5, we discuss our implementation of a d2d channel using LEDs and video camera.

2 Related Work

There exists a significant amount of prior work on the general topic of pairing. In their seminal work, Stajano, et al. [19] 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. Moreover, the infrared channels can not be perceived by humans and thus are easy to attack.

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 [5] by Levienet et al., Random Arts visual hash [10] 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 [8]. 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. [6], 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. Note that, however, the scheme is not applicable to pairing scenarios where one of the devices does not have a display or a speaker.

As an improvement to SiB [8], we earlier proposed a new scheme based on visual OOB channel [14]. The is the scheme that we analyze and improve upon in this paper. We will review this scheme in the following section and show that it is not secure in a security model in which the adversary has delaying/replaying capability on the OOB channel.

Uzun et al. [21] 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. Some recent work has focused upon pairing devices which possess constrained interfaces. These include the BEDA scheme [17], which requires the users to transfer the SAS strings from one device to the other using "button presses;" the schemes [11], [12], which require the users to compare simple blinking or beeping patterns on two devices. Most recently, the approach of [11] was extended by making use of an auxiliary device, such as a smartphone [15].

In [18], 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.

To summarize, the prior schemes are applicable to different pairing scenarios and have varying degree of usability. In this paper, our focus is on automated pairing methods using unidirectional OOB channels.

3 Communication and Security Model, and Applicable Protocols

We first review the communication and adversarial model for the SAS protocols as described in [22]. 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, drop, delay, replay and modify messages. On the OOB channel, the adversary can eavesdrop, drop, delay, replay and re-order messages, however, it can not modify them. In other words, the OOB channel is assumed to be an authenticated channel. Note that if two parties run multiple (serial/parallel) sessions with each other, then the adversary has the capability to delay, replay and re-order messages on the OOB channels among these sessions. We call such a model a "DRR-OOB" model.

We believe that considering a DRR-OOB model might be an overkill for certain OOB channels and certain applications, and that it would be useful to consider a weaker model where two parties never run parallel instances with each other and the adversary can eavesdrop and drop OOB messages, but it can not delay, replay and re-order them among multiple serial sessions between a pair of parties. We refer to such a model as an "nDRR-OOB" model. There might be various ways in which one can ensure such a model in practice. The easiest approach is to disallow a device to run more than one parallel session with a given party at a given time and whenever a new session is executed with the same party, have the device erase from its memory the old session with the same party.

The security notion for a pairing protocol is adopted from the model of authenticated key agreement due to Canneti and Krawczyk [2]. In the DRR-OOB model, we will consider an $(n, R, \bar{R})$-adversary $\mathcal{A}$ against the pairing protocol, which is allowed to launch only R sessions per player, and only $\bar{R}$ sessions between any *pair* of players. Note that in the DRR-OOB model, $\mathcal{A}$ is allowed to delay, replay and re-order OOB messages among multiple session between two parties, while in the nDRR-OOB model, $\mathcal{A}$ (which is effectively a $(n, R, 1)$-adversary) is not allowed to do so. In both these models, the security of the pairing protocol is modeled by an interaction between $\mathcal{A}$ and the challenger that operates the network of n players $P_1, ..., P_n$. In this game, the challenger has a *private input* of bit b. This 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.

Using the launch queries, $\mathcal{A}$ can trigger any of the n players P_i to start a session of the protocol with another player P_j. The challenger responds by initializing the state of the invoked session and sending back to $\mathcal{A}$ the message it generates. The adversary can also issue send queries for any previously initialized session on a message M as input, which triggers the challenger to deliver message M to that particular session and respond by following the protocol on its behalf. Moreover, on any of the launched sessions, $\mathcal{A}$ can also issue reveal query, which gives him the key output by that particular session, if this session computed a key, and a null value otherwise. Finally, on one of the sessions, $\mathcal{A}$ can issue a Test query. In response, if this session has not completed, the adversary gets a null value. Otherwise, if $b = 1$ then $\mathcal{A}$ gets the key output by the "tested" session, and if $b = 0$ then $\mathcal{A}$ gets a random BitString of the same length.

Eventually $\mathcal{A}$ outputs a bit $\hat{b}$. We say that an adversary has *advantage* ϵ in the attack, if the probability that $\hat{b} = b$ is at most $1/2 + \epsilon$. We say that the protocol is (T, ϵ)-**secure** if for all $\mathcal{A}$'s bounded by time T the above defined advantage of $\mathcal{A}$ is at most ϵ.

An example application of this model is during authentication for an ATM transaction, where there are only two parties, namely the ATM machine and a user, restricted to only three authentication attempts.

To date, two three-round pairing protocols based on short authenticated strings (SAS) have been proposed [9], [7]. These protocols all require bidirectional OOB channels and are proven $(T, nR\bar{R}2^{-k} + \epsilon)$-secure in the DRR-OOB security model. Of course, these protocols are secure in the weaker nDRR-OOB model as well. 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 3×10^{-4}, which gives us security equivalent to the security provided by 5-digit PIN-based ATM authentication [22].

4 Pairing with a Unidirectional OOB Channel

As we mentioned in the previous section, prior SAS protocols are proven secure in the DRR-OOB model, however they require bidirectional OOB channels. In this section, we focus upon pairing scenarios where bidirectional OOB channels do not exist.

We take a closer look at the protocol of [14], which requires a unidirectional OOB channel. We show the underlying protocol in Figure 1 (we base the protocol upon the SAS protocol of [9], although it can similarly work with other SAS protocols as well). The protocol works as follows[1]. Over the wireless channel, A and B follow the underlying SAS protocol. Then a unidirectional OOB channel is established by device A transmitting the SAS data. This is followed by device B comparing the received data with its own copy of the SAS data, and transmitting the resulting bit b of comparison over a OOB channel (say, displayed on its screen). Finally, the user reads the transmitted bit b and accordingly indicates the result to device A by transmitting the same bit b over an h2d input channel.

4.1 Protocol of [14] in the DRR-OOB Model

The protocol of [14] did not come with a security analysis [14]. Therefore, the first and a natural question is whether the protocol remains secure in a DRR-OOB security model. Unfortunately, the protocol turns out to be insecure in the DRR-OOB model. We show our attack next. In fact, we argue that it is hard to achieve pairing using only a unidirectional OOB channel in the DRR-OOB model.

The attack we describe next stems from the fact that only a single bit b, indicating a "success" or a "failure", is transmitted in the second step, i.e., over the d2h channel, and that this bit of information can be *delayed or replayed* (recall that our security model, described in Section 3, allows an adversary to do so!). Therefore, the attack is exploited as follows.

1. An instance of the above pairing protocol is run between two devices. The adversary does not insert any messages (specifically, its own public key(s), for which it knows the secret keys or its own secret shared key, based on the protocol) on the wireless channel. However, the adversary stalls the bit b (which indicates a "success") to be transmitted over the d2h channel. This forces the user to abort the protocol and re-run it.

[1] A similar modification was suggested to the protocol where devices exchange their public keys over the wireless channel and exchange the (160-bits long) hash of the concatenation of the two public keys over the OOB channel [14].

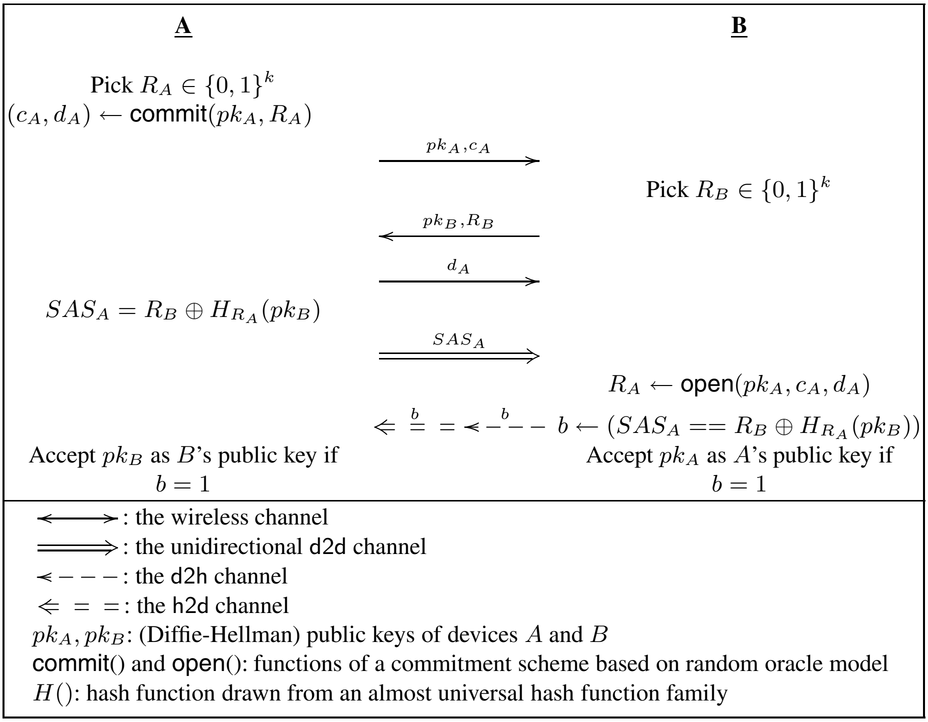

A
B
Pick $R_A \in \{0,1\}^k$
$(c_A, d_A) \leftarrow \mathsf{commit}(pk_A, R_A)$
pk_A, c_A
Pick $R_B \in \{0,1\}^k$
pk_B, R_B
d_A
$SAS_A = R_B \oplus H_{R_A}(pk_B)$
SAS_A
$R_A \leftarrow \mathsf{open}(pk_A, c_A, d_A)$
b
b
$b \leftarrow (SAS_A == R_B \oplus H_{R_A}(pk_B))$
Accept pk_B as B's public key if $b = 1$
Accept pk_A as A's public key if $b = 1$
: the wireless channel
: the unidirectional d2d channel
: the d2h channel
: the h2d channel
pk_A, pk_B: (Diffie-Hellman) public keys of devices A and B
commit() and open(): functions of a commitment scheme based on random oracle model
$H()$: hash function drawn from an almost universal hash function family

Fig. 1. The protocol of [14] based on the SAS protocol of [9]

2. During the second instance of the protocol, the adversary first inserts its own messages over the wireless channel and then delivers the previously stalled bit b over the d2h channel. Since the bit b indicates a “success”, the user is fooled into accepting the protocol instance instead of aborting it.

A similar attack can be based upon replaying of the bit b, instead of its delaying, as follows. Over the first instance of the protocol, the adversary does nothing except for recording the bit b. The adversary hopes that another instance of the protocol is run and if so, it attacks the new instance by inserting its own messages over the wireless channel, and simply replaying the previously recorded bit b over the d2h channel.

A general implication of the above attack on the protocol of [14] is that it seems hard, if not impossible, to achieve pairing with a unidirectional d2d channel. In other words, it appears hard to establish mutual authentication with a unidirectional d2d authenticated channel. We know, from the original unidirectional message authentication SAS protocol of Vaudenay [22], that a device A can authenticate itself to device B if there exists a physical channel from A to B. The question is can this SAS channel also be used by B to authenticate to A. Suppose that B wants to authenticate a message m_B to A. B can simply send m_B to A over the wireless channel, which the adversary might modify to m'_B. Now, both B and A need to know if m_B was modified during the transmission or not, i.e., if $m_B = m'_B$ or not. It is easy for B to know this: A can authenticate to B m'_B using the unidirectional SAS from A to B, and B can simply verify if $m'_B = m_B$ or not. However, there appears to be no way for A to know if it received the same message m_B that B transmitted, except for B itself notifying A whether or not $m'_B = m_B$, which can be achieved by B transmitting the bit b indicating the result of match over a "single-bit" authenticated channel from B to A. However, this brings us back to the attack that we described on the protocol of [14], since the bit b can be delayed or replayed by an adversary.

4.2 Protocol of [14] in the nDRR-OOB Model

As shown in the previous section, the protocol of [14] is not secure in the DRR-OOB model. Now, we analyze the protocol in the nDRR-OOB security model. As pointed out in Section 3, this is a more practical model for certain applications. Fortunately, the protocol can indeed be proven secure in the nDRR-OOB model. In fact, we show that using the modification as in the protocol of [14], any known SAS protocol P based on bidirectional OOB channels, which is secure in the DRR-OOB model (e.g., the protocol of [9]), can be converted into a pairing protocol Q based on a unidirectional channel in the nDRR-OOB model (e.g., the protocol of Figure 1).

Theorem 1. *If any known SAS protocol P is $(T, nR2^{-k}+\epsilon)$-secure against a $(n, R, 1)$-adversary in the DRR-OOB model, then the pairing protocol Q is $(T + \delta, nR2^{-k} + \epsilon)$-secure against a $(n, R, 1)$-adversary in the nDRR-OOB model, where δ denotes a small polynomial amount of time.*

Proof. We prove the above theorem by contrapositive. In other words, we show that if there exists a $(n, R, 1)$-adversary $\mathcal{A}$ in the nDRR-OOB model that can win against the protocol Q with a probability significantly better than $nR2^{-k}$ and in time T, then we can construct a $(n, R, 1)$-adversary $\mathcal{B}$ in the DRR-OOB model that can win against the protocol P with a probability significantly better than $nR2^{-k}$ and in approximately the same time T.

The idea of the construction of the adversary $\mathcal{B}$ is very simple. Basically, $\mathcal{B}$ receives the queries from $\mathcal{A}$, submits them to its challenger and forwards the responses received back to $\mathcal{A}$, thereby perfectly simulating the role of the challenger to $\mathcal{B}$.

$\mathcal{A}$ starts off by issuing the launch queries, which $\mathcal{B}$ submits to its challenger and responds back to $\mathcal{A}$ with the messages delivered by the challenger. $\mathcal{B}$ does the same when $\mathcal{A}$ issues the send queries for all the protocol messages on the wireless channel and the message transmitted over the OOB channel in one direction (say SAS_i from P_i to P_j) (since the two protocols are alike in terms of the messages exchanged over the wireless channel and message transmitted over the OOB channel in one direction). However, when $\mathcal{B}$ receives the message transmitted over the OOB channel in the other direction (say SAS_j from P_j to P_i) from the challenger, then $\mathcal{B}$ computes $b = (SAS_i == SAS_j)$ and sends b to $\mathcal{A}$.

When $\mathcal{A}$ issues the reveal queries and finally the test query, $\mathcal{B}$ simply submits these queries to the challenger and replies back with the responses it receives from the challenger.

If $\mathcal{A}$ succeeds in correctly distinguishing the key output by the "tested" session, so does $\mathcal{B}$; since the two protocols are exactly alike in terms of the messages exchanged over the wireless as well as the OOB channel in one direction, and also both essentially have the same winning condition, since $(b = 1) \Rightarrow (SAS_i = SAS_j)$. Both $\mathcal{A}$ and $\mathcal{B}$ win with the same probability and have only a small δ time difference that is needed by $\mathcal{B}$ in computing the boolean expression $b = (SAS_i == SAS_j)$.

5 Automated d2d Channel Using LEDs and Video Camera

In this section, we discuss our implementation of a d2d channel in which the transmitter is equipped with LEDs and the receiver is equipped with a video camera. Our channel implementation is quite efficient and its bandwidth, unlike the results of [14], improves with the increase in the number of LEDs. The implementation can also be used on regular displays by simulating the LEDs on them. In the pairing application, we use this channel to transmit 15-bits of SAS data.

5.1 Encoding Using LEDs

In our encoding, we need two types of LEDs: a "sync" LED for synchronization at the beginning and end of SAS data transmission, and one or more "data" LEDs for transmitting the SAS data. The sync LED is different in color from the data LEDs – in our setup we keep a red LED as the sync LED and green LEDs as the data LEDs. LEDs are placed horizontally and vertically on the transmitter display. The sync LED can be placed at any vertical or horizontal position. The bit locations of the data LEDs increases from left-to-right and top-to-bottom. So, the top left data LED shows the first bit of the SAS data. There should be some gap between the two LEDs which needs to be at least half of the width of the LED itself.

The sync LED is used for indicating the beginning and end of the SAS data transmission in order to detect any synchronization delays, adversarial or otherwise, between the two devices. The sync LED is kept in "ON" state only at the beginning and end of data transmission and in "OFF" state otherwise.

The data LEDs are used for SAS data transmission by indicating different bits ('0'/'1') for different states (OFF/ON) of LEDs; in our setup, we used the ON state of a data LED as a bit '1' and the OFF state as bit '0'. Each transmitter needs to have one or more data LEDs and the more in number of data LEDs are, the speedier the SAS data transmission becomes. The transmitter can send the number of bits equal to the number of data LEDs, i.e., one bit per LED at a time. If N is the number of Data LEDs, the transmitter can display N bits of SAS data at a time. For transmitting 15-bits SAS data it requires $\lceil \frac{15}{N} \rceil$ frames. The state of the sync and data LEDs is kept unchanged for a certain time period so that a stable state can be easily captured from the video stream of the receiver video camera. Each stable state captured from the video stream is termed as a "BitFrame". For our setup, the time duration a BitFrame is kept unchanged (henceforth referred to as the "hold time") is set to an experimentally determined value of 300 ms. After every 300 ms, next N bits of the SAS data are shown in the next frame. This process continues until all bits of SAS data are transmitted. If the last frame does not have N number of SAS bits to show, the first few LEDs show the data bits and the remaining are kept OFF.

For discovering the LEDs' location, color, dimension at the receiver side, we need two extra frames – an "All-ON" frame having all LEDs in ON state and an "All-OFF" frame having all LEDs in OFF state. Before transmitting the frames containing the SAS data, the All-ON and All-OFF frames are first displayed. These two frames are displayed within the same hold time of 300 ms. In addition to All-ON and All-OFF frames, we need another frame, to detect synchronization delays, having the sync LED in ON state and the data LEDs in OFF state. This frame is displayed at the end, after the completion of SAS data transmission. Therefore, overall we need a total of three extra frames. Thus, the total number of frames to be transmitted is $\lceil\frac{15}{N}\rceil + 3$, which yields a total transmission time of $(\lceil\frac{15}{N}\rceil + 3) \times 300$ ms, where N is the number of data LEDs.

5.2 Decoding Using a Video Camera

The two devices being paired first execute the protocol as in Figure 1 over the wireless channel. When the receiver device is done with SAS data computation, it turns on its video camera, asks the user of the device to adjust its camera setting, focus on the LED-based display of the transmitting device and press "OK" button when done. The user does the adjustment as needed and presses the OK button. After this, the receiver sends the "ready" signal to the transmitter and requests the transmitter to send the acknowledgement over the wireless channel when it is done with computing its SAS value and ready to start transmitting over the unidirectional channel. The transmitter acknowledges the receiver when it is ready for transmitting the SAS data and starts transmitting over the unidirectional channel. In this setting of unidirectional channel, the receiver must have higher reception rate than transmitter's transmission rate. So, the video camera must have higher frame rate than frame rate of the transmitters displayer. If frames are not carefully captured from the video stream, there is a chance of obtaining the counterfeit frames which contain the transition state of LEDs. Such frames may contain some LEDs of one state and some LEDs of next state.

Resolving the Timing Issue of Frame Capturing from Video Stream. Assuming that the transmission delay of acknowledgement from the transmitter to receiver is negligible (5-6 ms) compared to the "hold time" (of 300 ms) between two successive frames at the transmitter, the receiver captures the first frame from the video stream after a time equal to the half of the hold time (i.e., 150 ms) after receiving the acknowledgement. The receiver video camera also has a delay (about 30-40 ms, as most common cameras have a rate of 30-40 frames per second) of capturing the frame from video stream. So, the first frame is captured after the half of the hold time, after getting the acknowledgement from the receiver. The timestamps of capturing rest of the frames is pre-calculated by adding the hold time (300 ms) for each frames with capturing timestamp of the first frame. The captured frames are processed after the completion of capturing of all transmitted frames. During capturing, the captured frames are saved from the video stream buffer location of frames to another location in main memory for later processing. There is some initial delay in capturing of first frame and it is adjusted by capturing the frame in the middle of hold time, however, there is no delay per frame for capturing the rest of the frames. Thus, the frame capturing from the video stream works successfully in real time. Note that our scheme does not require global clock synchronization for the transmitter and receiver. Figure 2 depicts the synchronization of transmission and reception of data. In this figure each small rectangle on the receiving window denotes a

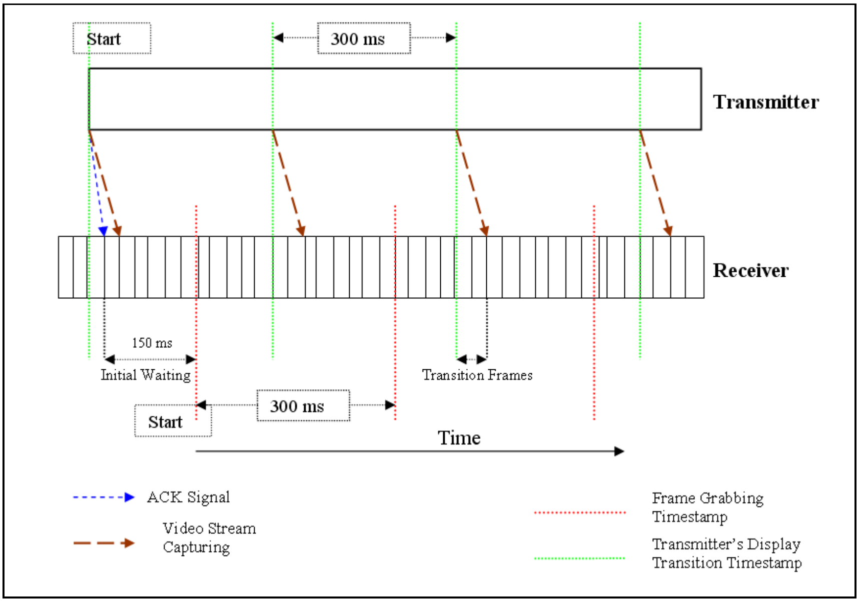


Fig. 2. Synchronization of Transmission and Reception of Data

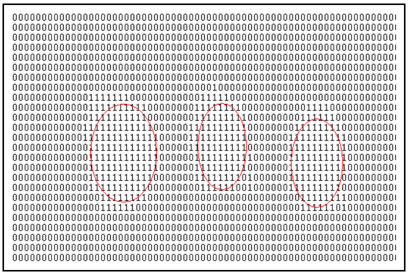
video frame of video stream and brown arrow marked with "Video Stream Capturing" denotes the propagation of transmitted signal to streamed frame in video stream, which makes sense that there is some propagation delay of an input transition from transmitter's side to receiver's video stream.

Detection of LEDs and Retrieval of SAS data from Video Frames. The frames are processed after the completion of capturing of all transmitted frames. The captured frames are processed by direct access to the memory address location of pixels. Direct addresses of pixels are calculated by knowing the pointer of memory address of first pixel of the frame and calculating other pixels address using the stride, width and length of frames.

Our LEDs location and dimension detection algorithm is a simple but fast, robust and efficient one - unlike any existing object/face detection algorithms [13,16,23]. It detects the position and dimension of LEDs deterministically. It is able to detect any shape/geometry of LEDs unlike [16,23] and doesn't require any prior training unlike [13,16]. It uses the color threshold adjustment technique like [24] to detect the LEDs position and dimension.

The maximal differences of RGB values, $max(dR, dG, dB)$ (denoted as μ), of each pixel of All-OFF and All-ON frames are measured and kept in memory, and using a threshold value for μ, BitStrings are built for each row of pixels. For example, if the μ exceeds a certain threshold, the corresponding bit in string becomes '1', otherwise it becomes '0'.

Each BitString is matched against a regular expression for consecutive 1s. For each matching, its center is calculated and its safeness and centeredness as an LED center is



```
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000010000000000000000000000000000000
000000000000011111110000000000001111100000000000000000000000000000
000000000000011111111110000000001111111100000000000111100000000000
000000000000011111111110000000011111111100000000001111110000000000
000000000000111111111111000000111111111000000000111111110000000000
000000000000111111111111100000111111111100000001111111110000000000
000000000000011111111111100000111111111100000001111111110000000000
000000000000011111111111100000011111111100000001111111110000000000
000000000000011111111111100000011111111100000001111111110000000000
000000000000011111111111000000011111111010000001111111110000000000
000000000000011111111110000000001111100100000001111111110000000000
000000000000000111111100000000000000000000000000011111110000000000
000000000000000111111000000000000000000000000000011111010000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
```

Fig. 3. Detected LEDs from BitString

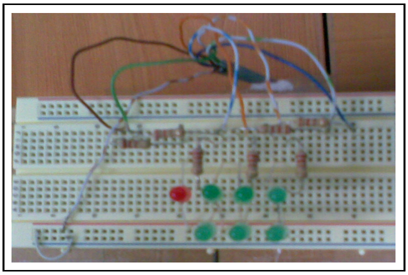


Fig. 4. Second Setting: breadboard with LEDs

checked by matching against the already explored LEDs and exploring only the adjacent pixels of this center in the frame. If its safeness and centeredness is proved, it is taken as an LED and put in explored list of LEDs. After checking for each match of regular expression for each bit strings, counts of explored LEDs is matched against the original count of the LEDs. This process continues up to a number of times by adjusting the threshold value of μ and constructing the new BitStrings until the count of LEDs matches. See Figure 3 for an example of detection of LEDs from the BitString.

After successful discovery of LEDs, the length, width, average RGB values of ON and OFF states of LED area for each LEDs are stored in memory for detecting the On-Off state of LEDs in subsequent BitFrames.

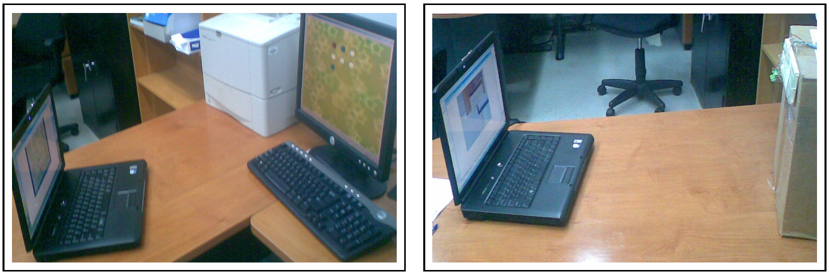
From the successfully discovered LEDs, red colored the sync LED is detected on the basis of its color. On the basis of the location of the sync LED, rest of the LEDs are clustered according to a threshold value of proximity among the LEDs. After successful detection of LEDs, the data LEDs are sorted according to the left-to-right and top-to-bottom ordering of coordinates for a maximum value of tolerance limit in deviation of coordinates. Tolerance limit is set by measuring the average width and length of discovered LEDs. Now for bit frames containing SAS data, average RGB values of the area of only the discovered LEDs are explored and matched against the ON and OFF state of RGB values of the LEDs with a tolerance limit. If the average RGB values of the area matches with the ON state of the LED, the corresponding bit is detected as ‘1’ and if these match with the OFF state, it is detected as ‘0’. In this manner, the whole SAS string is retrieved by exploring the discovered LEDs for each bit frames. If there arises any ambiguity, i.e., the average RGB values match with both the condition of ON and OFF state or neither of them, tolerance limit is adjusted and decoding is repeated up to a threshold number of times.

The last frame is examined to determine whether the sync LED is in the ON state and that all data LEDs are in the OFF state. If not, there is an indication of a sync bit failure due to synchronization delays.

If the extracted SAS matches with the computed SAS on the receiver and the frames pass the synchronization test, the receiver and transmitter are successfully paired. Otherwise, they fail due to mismatch of SAS or delay in synchronization. For a successful pairing, the LEDs are marked with a rectangle of green color around them and for a failed case, the LEDs are crossed with red color. Observing the graphical result on screen of the receiver, the user either accepts or aborts the pairing on the transmitter's device.

5.3 Experimental Setup

We implemented and tested our channel in two different settings. One showed the BitFrames on the monitor of a desktop PC and the other showed the BitFrames on real implementation of the scheme on breadboard using 7 LEDs (1 sync and 6 data LEDs), the breadboard being interfaced to a desktop PC using DB-25 parallel printer port. In the first setting, bitmap images of actual ON and OFF states of real LEDs are used. The pictures from the two settings are shown in Figures 5(a), 4 and 5(b).



(a) First Setting: transmitter is monitor

(b) Second Setting: transmitter is LEDs on breadboard

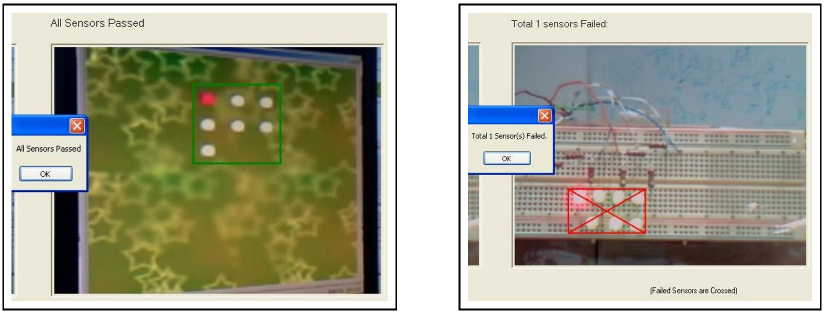
Fig. 5. Two Settings: receiver is laptop camera for both the settings

In both the schemes receiver is the Dell Vostro 1500 (Intel Core 2 Duo 1.6 GHz with 2 GB RAM) Laptop having Integrated Webcam and wireless channel is Wireless LAN (54 Mbps) of our university. The integrated webcam on the laptop has the capability of capturing 30 frames/second and it can take frames of fixed dimension of 640X480 pixels. We wrote the simulator in Microsoft VC# to implement the last two OOB message exchange as in the Figure of 1, assuming that the devices have already exchanged the first three messages over the wireless channel and each has computed its SAS value. The implementation processes the frames, extracts the SAS data from video stream and shows the result on screen. Actuation to real-timing is maintained by `Environment.TickCount` variable of MS VC#. The webcam is interfaced on port 1024 of the computer. Message passing is used to communicate with the webcam. We used "avicap32.dll" for capturing the video. The webcam can be replaced with any good IP Camera for better resolution of frames without any modification in the existing simulator. The camera is set in NON-STOP video capturing mode and frames are taken setting the camera in preview mode. Camera controller is added in the simulator to adjust the focus, tilt and pan of camera. For both the schemes transmitter is Dell Desktop (Intel Xenon Processor 1.8 GHz with 1 GB RAM). Communication with the transmitter is done by implementing the Client-Server communication model between the transmitter and receiver.

The simulator is used to generate huge number of test cases for different numbers and orientations of LEDs. We tested our system for different brightness of frames by setting different level of brightness of the monitor. In the first setting, frames are prestored in transmitter and are displayed when the receiver sent the request of transmitting the frames. It is used to test our pairing scheme on this channel easily and rigorously.

5.4 Experiment Results

A couple of snapshots of the results of execution of our scheme in both settings are shown in Figures 6(a) and 6(b).


All Sensors Passed

All Sensors Passed

Total 1 sensors Failed:

Total 1 Sensor(s) Failed.

(Failed Sensors are Crossed)

(a) First Setting: Successful Pairing (b) Second Setting: Failed Pairing

Fig. 6. Partial Screenshots of Results of the Two Settings

From the result of the first setting, we found that our implementation works for different number and orientation of LEDs. LEDs need not be in any fixed location on the screen and they can be on any random location while maintaining the left-to-right and top-to-bottom ordering among themselves. We tested lot of test cases for the different number and orientation of LEDs and all of them passed without any error. We also executed some test cases by changing SAS data on test cases, replacing the valid frames with invalid frames and replacing the sync frame with other frames. All these test cases were successfully determined as failing cases. By changing the brightness of the monitor, we found that our scheme works for any brightness of the monitor. Our first scheme works fine if the distance between the transmitter or receiver is less than 1.5–2 meters. If the distance is greater than 2 meter, the LEDs become to tiny to detect for the camera. In this case LED count doesn't match and the simulator searches up to a threshold number of times by adjusting the color threshold value. If it doesn't find any match of count of LEDs, it shows the failure message. By using a camera with a better resolution (which is currently 640X480) the distance of the transmitter and receiver can be increased.

Our implementation in the second setting also works fine. The Light from illuminating LEDs scatters around the LEDs. For testing the maximum light scattering effect we did not separate the LEDs by any type of separator or did not keep them inside the small holes to negate the lighting effect among the LEDs. All the test cases (for successful as well as failing pairing) passed without any problem. Our implementation of color threshold adjustment strategy during discovery of LEDs also works fine in this setting.

This setting is also tested with varying distances between the transmitter and receiver. It works fine up to 2.5 meters of distance, longer than in the first setting. Lights from LEDs directly falls on camera and thus ON-OFF state detection becomes easy as there is more change in lighting effect in this setting.

Based on the results obtained, we summarize the following salient characteristics of our implementation in both settings.

Transmission Time. Using N data LEDs and one sync LED, the transmission requires about $(\lceil \frac{15}{N} \rceil + 3) \times 300$ ms for 15-bit of SAS data. Extraction of SAS data from captured frames requires less than 1 second. Therefore, for a typical display which has 2 data LEDs and 1 sync LED, our scheme requires less than 4.15 s to complete the whole process. If a device has 5 data LEDs and 1 sync LED, it will require less than 2.8 s. Of course, since we need at least three extra frames, the transmission can not take place quicker than 900 ms, no matter how many LEDs we have. Moreover, we require a device to have at least two LEDs, one of which has a unique color.

Distance. Our scheme works well, if the distance between the sender and receiver is less than 1.5-2.5 meters. Real implementation of channel with the LEDs on breadboard in second setting shows that our channel is efficient for about 2-2.5 meter distance between the receiver and transmitter. This represents a promising improvement over the existing d2d channels which can work for very little distance between the transmitter and receiver.

Brightness and Intensity of Light. Our channel is robust to varying brightness and intensity of light. It compares the states of LEDs ON and OFF state on current default settings of brightness and intensity of light on devices we tested on. From first two, All-OFF and ALL-ON frames, it learns the environment.

5.5 Other Applications of Our Implementation

Our channel implementation using LEDs and video camera and our underlying algorithms for real-time capturing of the frames from video stream, processing the frames efficiently and extracting data from video frames, can be used in a number of applications. We briefly discuss some applications as follows.

Device Discovery. A device needs to, before starting to communicate with another wireless device, first determine its address. Currently, in Bluetooth and WiFi, such a device discovery is performed over the wireless channels and has serious implications in terms of efficiency as well as usability – the device senses for devices in the neighborhood and dumps a list of devices (which could be long) and asks the user to select the device it wants to connect to. Using our d2d channel, one can discover a device over the physical channel itself – the user goes near the device it wants to connect its device to and presses a button on it to start transmitting its address in the form blinking LEDs and reads the address using the camera. Using just two data LEDs, this would only take about 9 seconds to discover a 48-bit long Bluetooth device address, and ease the burden on the user.

Secure Key Distribution in Sensor Networks. Before, deploying a sensor network, the nodes need to be provided with keys that they can use to secure communicate among themselves. Due to the lack of a trusted infrastructure, such a key distribution needs to be performed on-site by the administrator of the network. Moreover, due to lack of hardware interfaces (such as USB interfaces) on sensor nodes and for usability reasons, the key distribution must be performed wirelessly. Using the existing, so called key pre-distribution schemes, e.g., [4], the key distribution can be achieved, if one could establish a secure channel or a key between each sensor node and the base station. We are currently in the process of extending our implementation of the d2d channel and the protocol of Figure 1 to simultaneously pair each sensor node with the base station. Most existing commercial sensor nodes generally have three LEDs and the base station PC can be easily connected with an IP camera. We claim that our approach would be much more efficient, scalable and user-friendly than a recently proposed scheme [3].

General Data Transmission. Our implementation can also be used in applications other than security, for data transmission. Using N data LEDs (or an equivalent sized display) and a hold time of 300 ms, we are able to achieve a bandwidth of $3.33N$ bps. With such a bandwidth, one could efficiently send out information such as advertisements, calendars, low-resolution images, etc.

6 Conclusion

In this paper, we focused upon pairing two devices using unidirectional OOB channels. We analyzed the protocol of [14] and proved its security in a reasonable security model. We also devised an efficient implementation of an OOB channel using LEDs as transmitter and video camera as a receiver. With a display consisting of just two LEDs, our implementations takes less than 6.5 seconds. With increase in the number of LEDs, the bandwidth of our channel gets better. For example, with six LEDs, we need less than 2.8 seconds. Our implementation has other useful applications in Bluetooth/WiFi device discovery, sensor network key distribution and in general for data transmission.

Acknowledgments. We would like to thank N. Asokan for his comments on an earlier version of this paper.

References

1. Balfanz, D., Smetters, D., Stewart, P., Wong, H.C.: Talking to strangers: Authentication in ad-hoc wireless networks. In: NDSS (2002)
2. 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. Springer, Heidelberg (2001)
3. Cynthia, K., Luk, M., Negi, R., Perrig, A.: Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes. In: ACM SenSys. (2007)
4. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A pairwise key pre-distribution scheme for wireless sensor networks. In: ACM CCS (2003)
5. Goldberg, I.: Visual Key Fingerprint Code (1996), http://www.cs.berkeley.edu/iang/visprint.c
6. Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and Clear: Human-Verifiable Authentication Based on Audio. In: ICDCS (2006)

7. Laur, S., Asokan, N., Nyberg, K.: Efficient mutual data authentication based on short authenticated strings. IACR Cryptology ePrint Archive: Report 2005/424 (2005)
8. 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 (2005)
9. Pasini, S., Vaudenay, S.: SAS-Based Authenticated Key Agreement. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958. Springer, Heidelberg (2006)
10. Perrig, A., Song, D.: Hash visualization: a new technique to improve real-world security. In: CrypTEC (1999)
11. Prasad, R., Saxena, N.: Efficient device pairing using human-comparable synchronized audiovisual patterns. In: Applied Cryptography and Network Security (ACNS) (2008)
12. 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)
13. Rowley, H.A., Baluja, S., Kanade, T.: Neural network-based face detection. In: Pattern Analysis and Machine Intelligence(PAMI) (1998)
14. Saxena, N., Ekberg, J.-E., Kostiainen, K., Asokan, N.: Secure device pairing based on a visual channel. In: IEEE Symposium on Security and Privacy (ISP 2006), short paper (2006)
15. Saxena, N., Uddin, M. B., Voris, J.: Universal device pairing using an auxiliary device. In: Symposium On Usable Privacy and Security (SOUPS) (2008)
16. Schneiderman, H., Kanade, T.: A statistical method for 3d object detection applied to faces and cars. In: TRINITY (2003)
17. Soriente, C., Tsudik, G., Uzun, E.: BEDA: Button-Enabled Device Association. In: International Workshop on Security for Spontaneous Interaction (IWSSI) (2007)
18. Soriente, C., Tsudik, G., Uzun, E.: Hapadep: Human asisted pure audio device pairing. Cryptology ePrint Archive, Report 2007/093 (2007)
19. Stajano, F., Anderson, R.J.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Security Protocols Workshop (1999)
20. Suomalainen, J., Valkonen, J., Asokan, N.: Security associations in personal networks: A comparative analysis. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572. Springer, Heidelberg (2007)
21. Uzun, E., Karvonen, K., Asokan, N.: Usability analysis of secure pairing methods. In: USEC (2007)
22. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
23. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition (2001)
24. Weszka, J.S.: A survey of threshold selection techniques. Computer Graphics and Image Processing 7, 259–265 (1978)

Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0.*

Nicolas T. Courtois[1] and Blandine Debraize[2,3]

[1] University College London, Gower Street, London, UK
[2] Gemalto, Meudon, France
[3] University of Versailles, France

Abstract. In this paper we analyse the algebraic properties over the field GF(2) of the addition modulo 2^n. We look at implicit quadratic equations describing this operation, and at probabilistic conditional linear equations. We show that the addition modulo 2^n can be partly or totally linearized when the output is fixed, and this for a large family of outputs. We apply these results to analyse the resistance of the stream cipher Snow 2.0 against algebraic attacks.

Keywords: Modular addition, multivariate quadratic equations, algebraic immunity, stream ciphers, Snow 2.0, algebraic cryptanalysis.

1 Introduction

Many ciphers are based on mixing of S-Boxes, arithmetic and Boolean operations. One of the fastest arithmetic operations is addition modulo a power of 2, that is handled very efficiently on modern processors. We will adopt the notation '$\boxplus$' for this operation in this article, to differentiate it from the "exclusive-or" which we denote '$\oplus$'. The addition modulo 2^n is used in block ciphers such as TwoFish, hash functions such as MD5 and SHA 1 and in many stream ciphers. Among stream ciphers, the most prominent example is Snow 2.0, which is today a reference standardized software-oriented stream cipher.

The algebraic immunity is defined as the minimum degree for which one can write multivariate equations mixing input bits and output bits (see [17]). It is an important design criterion for S-boxes. However this criterion is not sufficient in itself to define the resistance of an S-box against algebraic attacks. Indeed, the modular addition is partly linear, thus it has the same algebraic immunity as the exclusive or. Yet '$\boxplus$' is stronger. For example, an interesting analysis of the security of the stream cipher Snow 2.0 has been given in [5] by Billet and Gilbert. They propose an attack on a modified version of the cipher, where the '$\boxplus$' are replaced by '$\oplus$' that is fully linear. But as '$\boxplus$' cannot be described entirely by linear relations, the attack could not be directly extended to the real cipher.

A generalisation of the notion of algebraic immunity has been studied in [1] and [14]. It is concerned with implicit equations conditioned on the value of the output or a part of the output. It has interesting applications in algebraic cryptanalysis of stream ciphers as shown by Fischer and Meier in [14].

* Partly supported by the EU Commission via the ECRYPT network of excellence.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 328–344, 2008.

Algebraic properties of the addition modulo a power of two have previously been studied in [5,15,19]. In [5], a quadratic description over GF(2) implying carry bits is proposed. In this paper we propose a new notion called describing degree to explore the algebraic properties of the '$\boxplus$' in a more refined way. We develop a description of this function as a set of implicit quadratic equations over GF(2) without any additional variable. Then we study the question of how equations can be partially linearized from the point of view of the attacker. For this we use conditioned equations described in [14] to introduce a new method to approximate the addition modulo 2^n, that is the main contribution of this paper. Actually one can view '$\boxplus$' as an S-box in two different ways, and both versions can be partially or totally linearized. One of the interests of these partial linearization techniques is that they can improve considerably the complexities of algebraic attacks, that can potentially be developed also for other ciphers.

One of the propositions of the authors in [5] to extend their method to the real Snow 2.0 is to guess the carries of the modular addition. In this paper we go further in the analysis of the security of this cipher against this type of attack. We implement their proposition, and show that by using our linearization techniques on '$\boxplus$' we obtain better results than by guessing the carries. In this paper we consider KGSnow 2.0, that is the keystream generator part of the cipher, where the initial state of the registers is considered as the key of the cipher. We compare our results to the classical time-memory trade-off attack on KGSnow 2.0.

In Section 2 we recall and define various notions of algebraic immunity, in Section 3 we study algebraic deterministic descriptions of '$\boxplus$', and these can be partially linearized as shown in Section 4. In Section 5 we explain how to handle algebraic cryptanalysis and our basic algorithm. Based on all these, an analysis of KGSnow 2.0 is presented in Section 6.

2 Preliminaries

2.1 Notation

Let us consider three n-bit words $(x_{n-1},\dots,x_0)$, $(y_{n-1},\dots,y_0)$ and $(z_{n-1},\dots,z_0)$ with z_0 being the low-order bit. The modular addition

$$(x,y) \mapsto z = x \boxplus y \mod 2^n$$

is a T-function (see [16]), as each bit z_i of the output only depend on the bits $x_0,\cdots,x_i,y_0,\cdots,y_i$. This T-function can be described the following way by $(*)$ and $(*')$, using new variables that are carry bits, represented by the $(n-1)$-bit word $c = (c_{n-1},\dots,c_1)$:

$$(*)\begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + c_1 \\ z_2 = x_2 + y_2 + c_2 \\ \vdots \\ z_i = x_i + y_i + c_i \\ \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + c_{n-1}, \end{cases} \qquad (*')\begin{cases} c_1 = x_0 y_0 \\ c_2 = x_1 y_1 + (x_1+y_1)c_1 \\ \vdots \\ c_i = x_{i-1}y_{i-1} + (x_{i-1}+y_{i-1})c_{i-1} \\ \vdots \\ c_{n-1} = x_{n-2}y_{n-2} + (x_{n-2}+y_{n-2})c_{n-2} \end{cases}$$

2.2 Descriptive Algebraic Representation Criteria for S-Boxes

We first recall important notions in algebraic cryptanalysis:

Definition 1. *A system of equations is said to be overdefined if the rank of the system equations is strictly larger than the number of variables.*

Let $S : \{0,1\}^n \rightarrow \{0,1\}^m$ be an S-box.

Definition 2. *An I/O equation for S is a nonzero algebraic equation $r(x, y) = 0$ that holds with probability 1, i.e. for every pair (x, y) such that $S(x) = y$.*

The notion of Algebraic Immunity (also sometimes called Graph Algebraic Immunity or I/O degree) has been introduced by Carlet, Meier and Pasalic [17].

Definition 3. *The algebraic immunity AI is defined by the minimum degree of an I/O equation for S.*

The algebraic immunity of the modular addition is clearly 1, because of the linear equation mixing the least significant bits described at Section 2.1: $z_0 = x_0 + y_0$. The algebraic immunity of the exclusive or is also 1, yet typically $\boxplus$ will be cryptographically much stronger than $\oplus$. We see that the algebraic immunity is not always the best criterion to define the resistance of an S-box against algebraic attacks. Two other important properties of an S-box are:

1. The minimal degree d such that the S-box is entirely defined by equations of degree at most d.
2. The number of such linearly independent equations of degree at most d.

We define a new criterion to describe the first property :

Definition 4. *The minimal degree d such that the S-box is entirely defined by equations of degree at most d is called describing degree (DD) of S.*

The notion of algebraic immunity is based on the existence of an I/O degree equation. But if for a function F some equations of minimal degree d exist, however all these degree d equations may not define the function F. As we have seen, the algebraic immunity of '$\boxplus$' is 1 but as this function is not defined by the only one linear equation $z_0 = x_0 + y_0$, its describing degree is 2.

2.3 Criteria for Conditioned Algebraic Representation of S-Boxes

Conditional algebraic I/O equations emerge as an important tool in cryptanalysis of stream ciphers as illustrated by Krause, Armknecht, Fischer and Meier [1,14].

Definition 5. *Let us assume $n > m$. Given some fixed output y, a y-conditional I/O equation for S is a nonzero algebraic equation $r_y(x) = 0$ that holds with probability 1 for every x such that $S(x) = y$.*

The relevant notion of conditional algebraic immunity is defined by Fischer and Meier [14] as follows:

Definition 6. *Given some fixed output y, let d be the minimum degree of a y-conditional I/O equation. The conditional algebraic immunity CAI of S is the minimum of d over all y in $GF(2)^m$.*

Similarly, we adapt our describing degree criterion:

Definition 7. *Given some fixed output y, let d be the minimum degree such that the equation $S(x) = y$ is entirely defined by conditional I/O equations of degree at most d. The minimal d over all y in $GF(2)^m$ is called conditional describing degree (CDD) of S.*

3 Describing Degree of the Addition Modulo 2^n

In this section, we show that the addition modulo a power of two can be described by quadratic I/O equations over GF(2). We give a "describing" set of quadratic equations and compute many extra equations.

We note that there are several ways to consider $\boxplus$ as an S-box function. If we consider three n-bits words x, y and z, the equation

$$x \boxplus y = z$$

leads to two possible functions $\mathcal{P}$, $\mathcal{M}$ of type $\{0,1\}^{2n} \to \{0,1\}^n$:

- $\mathcal{P} : (x, y) \mapsto z$
- $\mathcal{M} : (x, z) \mapsto y$ and $\mathcal{M}' : (y, z) \mapsto x$ that is exactly the same function.

3.1 Equations with No Extra Variables

Proposition 1. *The Describing Degree is 2 for both $\mathcal{P}$ and $\mathcal{M}$.*

Proof. Looking at the equation $(*)$ and $(*')$ of Section 2.1, we notice that all the carry bits c_i from $(*')$, can be expressed as linear combinations of other variables using $(*)$, and eliminated. The resulting equations remain quadratic and there is no extra variable at all:

$$(\#) \begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + x_0 y_0 \\ z_2 = x_2 + y_2 + x_1 y_1 + (x_1 + y_1)(x_1 + y_1 + z_1) \\ \vdots \\ z_i = x_i + y_i + x_{i-1} y_{i-1} + (x_{i-1} + y_{i-1})(x_{i-1} + y_{i-1} + z_{i-1}) \\ \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + x_{n-2} y_{n-2} + (x_{n-2} + y_{n-2})(x_{n-2} + y_{n-2} + z_{n-2}) \end{cases}$$

At this stage, these equations entirely define the function $\boxplus$, but are not overdefined (this will change below). Yet they are already quite sparse. Apart from the linear terms, only products of type $A_i B_i$ or $A_i B_{i-1}$ do appear with A being x or y, and with B being x, y or z. The number of terms is only $\mathcal{O}(n)$ instead of $\mathcal{O}(n^2)$ for a generic system of quadratic equations.

3.2 Additional Equations

Many other quadratic equations do exist for the adders modulo 2^n. Their existence can be derived as follows:

- For any n, from (#) we have 1 linear and $n-1$ quadratic equations.
- Then there are $3n$ additional equations that come from the fact that you can multiply the linear equation by any variable and it becomes quadratic. However it turns out that the dimension of the vector space spanned by these equations is only $3n-1$ because of the following linear dependency: $(z_0+x_0+y_0) = z_0(z_0+x_0+y_0)+x_0(z_0+x_0+y_0)+y_0(z_0+x_0+y_0)$
- We also have 2 equations that come from the fact that you can multiply $z_1 = x_1+y_1+x_0y_0$ by x_0 or by y_0.
- Then for each of the $n-2$ remaining equations, one gets 3 additional quadratic equations. This is because the equation:
$$\begin{aligned} z_i &= x_i + y_i + x_{i-1}y_{i-1} + (x_{i-1}+y_{i-1})(x_{i-1}+y_{i-1}+z_{i-1}) \\ &= x_i + y_i + x_{i-1} + y_{i-1} + x_{i-1}y_{i-1} + x_{i-1}z_{i-1} + y_{i-1}z_{i-1} \end{aligned}$$
can be multiplied by $(x_{i-1}+y_{i-1})$, by $(x_{i-1}+z_{i-1})$ and also by $(y_{i-1}+z_{i-1})$. These three new equations are linearly dependent and their rank is two. Thus we get $2(n-2)$ additional equations.

In the extended version of this paper, we prove that all the $6n-3$ quadratic equations described above are linearly independent.

4 Conditional Linear Equations for '$\boxplus$'

By fixing z for $\mathcal{P}$, and y for $\mathcal{M}$, we obtain conditional equations of degree at most 2, with at least 2 linear equations. In some cases, when the $n-2$ least significant bits of z are 1 for $\mathcal{P}$, and when the $n-2$ least significant bits of y are 0 for $\mathcal{M}$, it can be completely linearized. That is what we show in section 4.1.

At section 4.2, we refine this result by showing that for both $\mathcal{P}$ and $\mathcal{M}$, if the output contain r consecutive bits of the same value 0 or 1 (whatever the value it is for both $\mathcal{P}$ and $\mathcal{M}$), we obtain a set of $r+2$ linear equations that are *simultaneously* true with a certain probability.

4.1 Conditional Describing Degree of '$\boxplus$'

Proposition 2. *The Conditional Describing Degree is 1 for both $\mathcal{P}$ and $\mathcal{M}$.*

Proof. For $\mathcal{M}$, this result is straightforward: we put $y = 0$, and the relation becomes completely linear as we have $x = z$.
For $\mathcal{P}$, we put $z = 2^n - 1$, and we see that the first carry $c_1 = 0$, because $x_0 \oplus y_0 = 1 \Rightarrow x_0 \boxplus y_0 < 2$. Then we prove recursively that all the carries of this function are zero and that: $\forall i \; x_i \oplus y_i = 1$. Finally, these equations clearly describe exactly all possible input values that lead to the chosen fixed output for this S-box (for both $\mathcal{M}$ and $\mathcal{P}$).

In fact $y = 0$ is not the only value for y such that the relation $\mathcal{M}(x, z) = y$ can be entirely described by linear equations. Our simulations showed that exactly 4 values of y have this property: the values for which the $n-2$ least significant bits of y are zero. This can be proven as follows : as the carry c_{n-2} is 0, the equation of the second most significant bit of the modular addition is $z_{n-2} = x_{n-2} \oplus y_{n-2}$. If $y_{n-2} = 0$, the same result holds for the most significant bit equation: $z_{n-1} = x_{n-1} \oplus y_{n-1}$. If $y_{n-2} = 1$, we have: $c_{n-1} = \lfloor \frac{x_{n-2}+y_{n-2}+c_{n-2}}{2} \rfloor = \lfloor \frac{x_{n-2}+1}{2} \rfloor = x_{n-2}$. Then the most significant bit equation is $z_{n-1} = x_{n-1} \oplus y_{n-1} \oplus x_{n-2}$.

The same way, $z = 2^n - 1$ is not the only value for z such that $\mathcal{P}(x, y) = z$ can be fully described by linear equations. Our simulations showed that the 4 values of z such that its $n-2$ least significant bits are 1 have the same property. This can be proven in a similar way as for $\mathcal{M}$.

When we fix the output, the number of linear equations describing $\mathcal{P}$ and $\mathcal{M}$ is at least 2 and in many cases it is more than 2. One can observe that (this point will be developed later) :

- For $\mathcal{P}$, this depends on the number of consecutive 1 in the least significant bits of the binary representation of z. If there are r consecutive 1 and $r \leq n-2$, the number of linear equations is $r+2$.
- For $\mathcal{M}$, this depends on the number of consecutive 0 in the least significant bits of the binary representation of y. If there are r consecutive 0 and $r \leq n-2$, the number of linear equations is $r+2$.

4.2 Probabilistic Conditional Properties of '$\boxplus$'

We will now give two general theorems on the number of probabilistic conditional equations for $\mathcal{P}$ and $\mathcal{M}$. Let $S : \{0,1\}^n \rightarrow \{0,1\}^m$ be an S-box.

Definition 8. *A set $\mathcal{E}$ of equations is said to be p-probable for S if the probability that all equations in $\mathcal{E}$ are* simultaneously *true, taken over the set of all pairs (x, y) such that $S(x) = y$, is equal to p.*

Theorem 1. *Let z be a fixed output for $\mathcal{P}$.*

- *If z has r consecutive 1s from the bit $i \geq 0$ to the bit $i+r-1 \leq n-1$ in its binary representation, then there is a p-probable set $\mathcal{E}$ of $r+2$ (only $r+1$ if $i+r-1 = n-2$, and r if $i+r-1 = n-1$) linear equations for $\mathcal{P}$, with $p = \frac{1}{2} + \frac{1}{2^{i+1}}$.*
- *If z has r consecutive 0s from the bit 0 to the bit $r-1 \leq n-1$ in its binary representation, then there is a p-probable set $\mathcal{E}$ of $r+2$ (only $r+1$ if $r-1 = n-2$, and r if $r-1 = n-1$) linear equations for $\mathcal{P}$, with $p = \frac{1}{2}$.*
- *If z has r consecutive 0s from the bit $i \geq 0$ to the bit $i+r-1 \leq n-1$ in its binary representation, then there is a p-probable set $\mathcal{E}$ of $r+2$ (only $r+1$ if $i+r-1 = n-2$, and r if $i+r-1 = n-1$) linear equations for $\mathcal{P}$, with $p = \frac{1}{2} - \frac{1}{2^{i+1}}$.*

A symmetrical result holds for $\mathcal{M}$ when replacing 0s by 1s:

Theorem 2. *Let y be a fixed output for $\mathcal{M}$.*

- *If y has r consecutive 0s from the bit i to the bit $i + r - 1 \leq n - 1$ in its binary representation, then there is a p-probable set $\mathcal{E}$ of $r + 2$ (only $r + 1$ if $i + r - 1 = n - 2$, and r if $i + r - 1 = n - 1$) linear equations for $\mathcal{M}$, with $p = \frac{1}{2} + \frac{1}{2^{i+1}}$.*
- *If y has r consecutive 1s from the bit 0 to the bit $r - 1 \leq n - 1$ in its binary representation, then there is a p-probable set $\mathcal{E}$ of $r + 2$ (only $r + 1$ if $r - 1 = n - 2$, and r if $r - 1 = n - 1$) linear equations for $\mathcal{M}$, with $p = \frac{1}{2}$.*
- *If y has r consecutive 1s from the bit $i \geq 0$ to the bit $i + r - 1 \leq n - 1$ in its binary representation, then there is a p-probable set $\mathcal{E}$ of $r + 2$ (only $r + 1$ if $i + r - 1 = n - 2$, and r if $i + r - 1 = n - 1$) linear equations for $\mathcal{M}$, with $p = \frac{1}{2} - \frac{1}{2^{i+1}}$.*

For proofs of Theorem 1 and 2 we refer to Appendix A.

Remark. In both theorems 1 and 2, the assertions are true with probability 1 if we fix another constraint. For assertion 1 and 3, this constraint consists in fixing the carry c_i to 0. For assertion 2 of Theorem 1, it consists in fixing x_0 to 0, and for assertion 2 of Theorem 2 , in fixing x_0 to 1 (this also implies that the first carry bit c_1 is 0). This can be seen in the proof in Appendix A. A randomly chosen linear equation is true for $\mathcal{M}$ or $\mathcal{P}$ with probability close or equal to $\frac{1}{2}$. The interest of our theorems is that they allow to obtain r, $r + 1$ or $r + 2$ linear equations that are *simultaneously* true with probability close to $\frac{1}{2}$. Thus in cryptanalysis, one is able to, if certain constraints are added, to partially linearize the modular addition. Adding a large number of linear equations to a quadratic system allows to eliminate many variables, and is expected to make it in general easier to solve, at least for known Gröbner bases algorithms. This concept of simultaneous linear approximations is very different from Linear Cryptanalysis with multiple characteristics, and to the best of our knowledge has not been studied before.

5 Algebraic Cryptanalysis and Application to KGSnow

5.1 Overview of Algebraic Cryptanalysis

Algebraic cryptanalysis is usually made of two stages:

1. **Writing the equations.** describing the problem of the recovery of the key. This stage determines the complexity of the second stage.
2. **Solving the polynomial system**. The most usual family of algorithm for solving polynomial systems is the XL and Gröbner bases family. XL ([7]) and Gröbner bases algorithms like F4 ([12]) and F5 ([13]) are based on the same idea of *expansion* of the system followed by an *elimination* step.
   - During the *expansion* the equations are multiplied by monomials of a chosen degree.

- The *elimination* is a Gaussian elimination applied to the expanded system where each monomial is considered as a variable.

This principle is applied once in XL whereas it is applied several times for F4 and F5 with additional clever tricks to decrease the number of equations in the Gaussian elimination.

The theoretic bound for the complexity of XL,F4 and F5 algorithm depends on the maximal degree d of the polynomials manipulated during the computation of the algorithm. The most important cost in these algorithms is the complexity of the (most costly step) Gaussian elimination, that is bounded by $\mathcal{M}^3$, where $\mathcal{M} = 1 + n + \binom{n}{2} + ... + \binom{n}{d}$ is the number of monomials of degree less than d (n being the number of variables).

Some work has been done to compute the value of this degree for a given system by Diem in [10] and Bardet, Faugère and Salvy in [4]. But it is not relevant for non random systems such as the systems derived from cryptographic systems. What is well-known and proved by [4] is that the more overdefined the system is, the lower this degree is. Thus, experimentation plays still an important role in evaluating which overdefined systems of equations derived from cryptographic systems can be solved and how.

5.2 Description of ElimLin and Simulations on KGSnow 2.0

In our experiments on KGSnow 2.0. we are heavily limited by the computational power available and in this paper we limit the maximal degree for the polynomials used during our computations to a very small value that is 2. We handle all our computations with a very simple algorithm for solving polynomial systems over GF(2) that is called ElimLin. A high-level description of ElimLin is given in Algorithm 5.1. It is hard to make a fast and memory efficient implementation of this algorithm, and serious research is needed about how to handle sparse Gaussian elimination and how do we store equations in memory in ElimLin. It

Algorithm 5.1. ElimLin algorithm.

```
INPUT: a system $\mathcal{S}$ of GF(2)-equations $\{p_1, ..., p_m\}$ describing an ideal $\mathcal{I}$
Apply a total order on the monomials of $\mathcal{S}$
$\mathcal{S} \longleftarrow$ *Gaussian elimination*$(\mathcal{S})$
$L \longleftarrow$Number of linear equations in $\mathcal{S}$
while $L > 0$:
    for $i = 1$ to L:
        $v \longleftarrow$ greatest variable of the linear equation $l_i$
        $l'_i \longleftarrow l_i \oplus v$
        Substitute $v$ by $l'_i$ in all the equations of $\mathcal{S}$ except from $l_i$
    Apply a total order on the monomials of $\mathcal{S}$
    $\mathcal{S} \longleftarrow$ *Gaussian elimination*$(\mathcal{S})$
    $L' \longleftarrow$ Number of linear equations in $\mathcal{S}$
    $L \longleftarrow L - L'$
return $\mathcal{S}$
```

appears that (with our current version already) we do not obtain better results with the F4 version of the computer algebra system MAGMA (see [21]) than with the simple ElimLin.

ElimLin in Cryptanalysis of KGSnow 2.0

In our simulations on KGSnow 2.0 we write the I/O equations describing the update of the LFSR and FSM, and the output of the cipher occurring for some consecutive clocks (from 11 to 18 consecutive clocks). The equation describing the addition modulo 2^{32} are those described at section 3. The equations describing the 32 bits S-box are directly derived from the 39 I/O quadratic equations describing the AES S-box (see [8]). We fix some bits belonging to the states of the FSM, and apply ElimLin on this system.

We consider here that brute force is the exhaustive search of the LFSR and FSM initial states (576 bits). If we fix all but a key bits, an attack will be faster than brute force if the running time is less than $2^a E$, where E is the time to check one potential possibility for the initial state. Given a sufficient number of output bits, heuristically about $|LFSR| + [R1| + |R2| + \varepsilon$, (as in [2]), this system has a unique solution that gives these 'key' bits. Exact figures are hard to evaluate because they depend on an optimised implementation of the cipher. Here we will assume that one encryption takes 300 CPU clocks and that the CPU runs at 3 GHz. Then $E \approx 2^{-35}$ hours. Thus, if we fix all key bits except 35, an attack done in less than 1 hour on a PC will be faster than brute force. If we fix all except 40 key bits, any attack done in less than (approximately) 1 day, will be faster than the exhaustive search of the initial LFSR and FSM bits.

6 Analysis of Snow 2.0 and KGSnow 2.0

KGSnow 2.0 is the keystream generator part of Snow 2.0. In this part we give a short description of Snow 2.0 and recall the analysis of the security of this cipher given in [5] and propose new ways to investigate this security by studying KGSnow 2.0 and using our results of sections 3 and 4.

6.1 Description of Snow 2.0

Snow 2.0 is a reference standardized software-oriented stream cipher. It is believed quite secure and is quite fast: less than 6 CPU cycles per byte on a PC, which is roughly about 3 times faster than RC4 and 4 times faster than AES, cf. [20]. We give a brief description of Snow 2.0; see [11] for details. Snow 2.0 stream cipher is based on one linear feedback shift register made of 16 elements from $GF(2^{32})$ (that can also be seen as a binary LFSR with 512 bits), and a finite state machine composed of 2 states of 32 bits each, nonlinearly clocked.

The output of the FSM is given by the equation:

$$z_t \oplus s_t = (s_{t+15} \boxplus R1_t) \oplus R2_t \tag{1}$$

and the update of the FSM is given by:

$$R1_{t+1} = s_{t+5} \boxplus R2_t \tag{2}$$

$$R2_{t+1} = S(R1_t), \tag{3}$$

where S represents the S-box. This 32 bits S-box is made of four parallel Rijndael S-boxes followed by the MixColumn operation (see [9]). The Rijndael S-box can be described by 39 I/O quadratic equations, see [8]. The MixColumn transformation is GF(2) linear then the entire S-box can be described by 156 quadratic I/O equations. The length of the key is 128 or 256 bits and the initialisation of the key is nonlinear. Our contribution, in section 6.3 essentially consists in analysing KGSnow 2.0.

KGSnow 2.0. has the same design as Snow 2.0, but we ignore the key and IV setup, this meaning that we consider that the key of KGSnow 2.0. is the initial state of the registers when the first keystream bits are produced. This is because with algebraic attacks, the values we are looking for are the values of these registers as they are the solution of the systems of equations, and especially because we do not even know today concerning the security of Snow 2.0 against algebraic attacks if these attacks are able to recover this state faster than its exhaustive search.

6.2 Previous Work

The best known attacks on Snow 2.0 are distinguishing attacks. In [18], it is shown that it possible to distinguish an output keystream of Snow 2.0 of length 2^{174} words from a truly random sequence with workload 2^{174}. No key recovery attack on this cipher have been found so far.

In [5], Billet and Gilbert analyse the security of the cipher by replacing the addition modulo 2^{32} by '$\oplus$'. It is then possible to break the modified cipher by linearization with a complexity of 2^{51}. They use the fact that the describing degree of the S-box is 2, added to the fact that with the replacement of the '$\oplus$' by '$\boxplus$', it is possible to eliminate all the FSM memory bits except the initial ones. They proposed two ways to exploit this result for the real Snow 2.0.:

- The first one consists in guessing the carries of the addition modulo 2^{32}, or to look for the most probable case, that is when all the carries are 0. We have implemented this approach by fixing some carries in the system of equations. But by applying our algorithm (see section 5.2) on such systems we could never recover the initial state fast enough to perform an attack more efficient than the exhaustive search. In section 6.3 we improve this method by using our results of section 4.1.
- The second one consists in introducing the carry bits of the two modular additions '$\boxplus$' at each clock. This allows to build a system of quadratic equations describing the initial state. But in this case it is not possible anymore to totally linearize the system of equations, the attacker has to apply a polynomial system solving algorithm like Gröbner bases algorithms to recover

the initial state. We give an overview of this kind of algorithm and their complexities in Section 5. This second approach can be much improved by our analysis of the modular addition proposed in part 3. Indeed, as we explain in Section 5, the more overdefined the polynomial set is, the better the complexity of solving the system becomes. The equations produced by our method implies exactly the same number of variables and provides $189n$ quadratic equations for each $\boxplus$ instead of $31n$. Another advantage comes from the fact that these equations are very sparse.

As the theory is very poor concerning the complexity of algorithms like Gröbner Bases algorithms, the effectiveness of this kind of attacks remains extremely unclear. An interesting question that has not been answered so far concerning this type of attack is the following: is an algebraic attack able to break KGSnow 2.0? We show at section 6.3 that by using the algebraic properties of '$\boxplus$', the answer is yes.

6.3 Towards an Optimal Linearizing Attack

In [5], the authors propose to linearize the $\boxplus$ by guessing the carries. We fixed $17 \times 31 + 17 \times 31$ carries of consecutive clocks and applied ElimLin on the system of the linearized equations coming from the 34 $\boxplus$ and the quadratic I/O equations describing the 17 uses of the S-box. We suppose that when the carries are fixed, the solving part behaves the same way as if all the carries were zero. The probability for this event to happen for 16 consecutive clocks, as proposed in [5], is : $(\frac{3}{4})^{31*17}(\frac{2}{3})^{31*15} \simeq 2^{-497}$. (The approximation of this probability in [5] is too large, as all the random variables are not independent). ElimLin had not finished its computations after 80 hours. Then this attack is not faster than the exhaustive search of the initial state.

We show in this part that guessing the consecutive values of the register R1 and using the properties of '$\boxplus$' described at section 4 seems to be a much better strategy. This is our contribution. We use here the word 'linearizing' to differentiate from linearization attacks where the term linearization means considering each monomial as a variable. Here the concept is completely different, as we do not increase the number of variables.

First Attack. We observe that in the design of KGSnow 2.0 (this coming from the design of Snow 2.0) , the fact that for a given t, $R2_{t+1}$ only depends on $R1_t$: by guessing $R1_t$, one gets immediately $R2_{t+1}$. Then, by guessing several consecutive values of $R1$, we obtain also several consecutive values of $R2$. With this method we still do not completely linearize the equations: each time the value of $R1_t$ is known, by equation 1 we just know that we get at least 2 linear relations, the other ones remaining quadratic. But each time the values of $R2_t$ and $R1_{t+1}$ are simultaneously known, we obtain the values of 32 bits of the internal state. The known values and linear relations between the internal state bits are also linear relations between the bits of the initial state of the LFSR as each bit of the internal sequence can be expressed as a linear expression of the initial LFSR state bits. The same way, the quadratic relations between the internal state bits means that we have quadratic equations between bits of the initial state of the LFSR as a composition $f \circ l$ where f is a degree 2 boolean function and l a linear boolean function is still a degree 2 function.

If we guess 10 consecutive $R1s$ (320 bits to guess), we obtain an overdefined system of linear equations and quadratic equations implying only the bits of the 512 initial state bits of the LFSR. These equations come from 17 additions modulo 2^{32}. As each '$\boxplus$' provides an amount of information of 32 bits, the information provided by this quadratic and linear boolean function is enough to recover the 512 initial state bits of the LFSR.

By applying any Gröbner bases algorithm on this system of equations, we obtain the initial state bits. No method is known to compute compute a theoretical complexity for this multivariate polynomial system solving part. Under the hypothesis that this system can be solved at degree 2, a theoretical complexity would be $\mathcal{O}(2^{51})$. In practice we could solve such systems with an algorithm called ElimLin described in Section 5.2 in 2.4 minutes on a PC, which is much faster than the theoretical complexity.

Even if this method seems much better than the guess of the carries, it does not completely linearize the equations. Actually we are able to do it by using the conditional properties of '$\boxplus$' described at Section 4.2.

Improvement of the Attack. In this part the idea is to go through the keystream to look for the most interesting case instead of guessing the information. We will use the facts described in theorem 1 and theorem 2 that $11...11 = 2^{32} - 1$ being an output of $\mathcal{P}$ and $\mathcal{M}$ simultaneously linearize both operations with almost the same probability, (as $11...11 \boxplus 1 = 00...00 \mod 2^{32}$), and the trivial fact that 0 being an output of $\mathcal{P}$ linearizes the operation.

We look for the case when $R1_1 = 0, R1_2 = 2^{32} - 1, R1_3 = 0, R1_4 = 0, R1_5 = 0, R1_6 = 0, R1_7 = 0, R1_8 = 0, R1_9 = 0$ by going through the keystream. These constraints on the $R1_i s$ are essentially constraints on the internal sequence: we need that

- $s_5 = -R2_0 \mod 2^{32}$,
- $s_6 = -1 - R2_1 \mod 2^{32}$,
- $s_7 = -S(0) \mod 2^{32}$,
- $s_8 = -S(2^{32} - 1) \mod 2^{32}$,
- for $9 \leq i \leq 13$, $s_i = -S(0) \mod 2^{32}$.

As the LFSR is clocked by a primitive feedback polynomial, the period of the internal sequence is $2^{512} - 1$ and the probability for this constraint to happen is 2^{-288}. We know that we are at the right place on the keystream when the system of equations is solved and provides the right initial state of the LFSR.

Let us suppose these constraints are verified. The same way as the previous attack, we obtain 224 GF(2) linear relations implying only initial LFSR state bits by equation (2). Each time $R1$ is 0, from clock 3 to 9, we obtain 32 GF(2) linear relations in the initial LFSR state bits by equation (1), that is an amount of 224 linear equations. By equation (2) at clock 1 we have:

$$s_6 \boxplus R2_1 = 111 \cdots 1.$$

Then from the proof of proposition 4.1 we obtain :

$$R2_1 = s_6 \oplus 111\cdots 1.$$

By replacing $R2_1$ by $s_6 \oplus 111\cdots 1$ and $R1_1$ by 0 in equation (1), we obtain 32 new GF(2) linear relations in the initial LFSR state bits. Finally by using Theorem 2 (Section 4) for equation (1) at clock 2, we obtain 32 linear relations in the LFSR initial state bits with probability $\frac{1}{2}$. To obtain a probability 1, it is enough to add another constraint on the internal sequence: we need that the least significant bit of $s_2 \oplus z_2 \oplus S(0)$ is zero. This is because we have by equation (1) at step 2:

$$s_2 \oplus z_2 \oplus S(0) = s_{17} \boxplus (2^{32} - 1),$$

then, as in proof of theorem 1, we obtain:

$$(s_2 \oplus z_2 \oplus S(0)) \boxplus 1 = s_{17}.$$

If the least significant bit of $s_2 \oplus z_2 \oplus S(0)$ is zero, there is no carry with probability 1 in this modular addition. As z_2 is known, it is a constraint on the least significant bit of s_2.

The total number of linear equations is 512. If we assume that these equations are linearly independent, the system can be solved by a simple Gaussian reduction. But this Gaussian reduction can be performed as a precomputation step: during the precomputation the keystream bits are not known and become variables. We then obtain each initial LFSR state bit as a linear combination of the 9×32 keystream bits variables that are used in the 512 equations described above. Then the final step consists in replacing the keystream variables by their real values and verifying that the LFSR initial state bits are correct. This would give a final time complexity of about 2^{288}.

If the rank of the system is $r < 512$, we precompute r initial state bits as linear boolean functions of the keystream variables and the last $512 - r$ initial state variables. The complexity is then multiplied by 2^{512-r} as we have to guess the last $512 - r$ variables. Actually our simulations showed that this rank is 504. We show in Appendix B, that by using theorem 1, it is possible to compute at least two more linear equations in the LFSR initial state bits. The final time complexity of this attack is then at most 2^{294}.

The drawback of this method compared to the first one we have proposed is of course the huge amount of necessary keystream bits, about $2^{288} \times 32 = 2^{293}$. This amount can be reduced because, as we explain in section 4, fixing only the 30 least significant bits of the 9 states $R1_t$ have the same linearizing properties as fixing all the bits. We just guess the 2 most significant bits of $R1_1, \cdots, R1_9$ instead of looking for their right value. We store a 2^{18} entries lookup table in which we set the values of the key depending on the keystream variables for each value of the 18 bits coming from the 2 most significant bits of each $R1_1, \cdots, R1_9$. The time complexity is the same but the keystream requirements are lower, about 2^{275} bits. The space complexity is quite small, (about 16 Gb to store the table).

In [3], a time-memory trade-off is proposed, with $TM = N$ and $D = T$, where T is the time complexity, M the memory, D the data and N the number of possible states. We can compare our method with this attack on KGSnow 2.0 with the precise parameters of our method. We show in Table 1 that we obtain better results with our attack. In [6] a better time-memory-data trade-off is proposed: $N^2 = TM^2D^2$, with the following constraint: $D^2 \leq T$. Because of this constraint, in principle it is impossible to compare this attack with ours.

Table 1. Our attack on KGSnow 2.0 vs. general time-memory trade-off

	time	memory	keystream
Babbage time-memory trade-off	2^{302}	2^{295}	2^{287}
Our best attack	2^{294}	2^{37}	2^{275}

Remark. We have made some computations on KGSnow 2.0 to try to improve this attack by guessing fewer variables. We have written the system of equations describing 9 consecutive clocks of KGSnow 2.0. In this system we fixed all the bits of $R1_1$, $R1_3$, $R1_4$, $R1_5$, $R1_6$, $R1_7$, $R1_8$, $R1_9$ to 0 except from the most significant bits, and the five most significant bits of $R1_9$, and we fixed the 31 least significant bits of $R1_2$ to 1. We were able to recover the initial LFSR state bits in 2.08 hours. The total complexity is here 2^{311} (see Section 5.2 for details on the computation of the total complexity). We could not obtain a better complexity than 2^{311} by fixing less variables, this meaning that we were not able to improve the complexity of the theoretical attack described above.

7 Conclusion

In this paper we study multivariate linear and quadratic properties over GF(2) of the addition modulo 2^n. We propose a new method for describing this operation as an overdefined system of implicit boolean equations of degree 2. We also introduce the concept of multiple and simultaneous linear approximations. This concept is different from Linear Cryptanalysis with multiple characteristics: we show how to partially or completely linearize the boolean equations describing this function by setting appropriate specific constraints on the output or/and one of the inputs.

These properties can be used to design conditional linearizing attacks on ciphers that use additions modulo 2^n. We propose an example of application in cryptanalysis of KGSnow 2.0, the keystream generator part of Snow 2.0. Given the specific structure of KGSnow 2.0, we have found a combination of constraints, such that the number of linear equations obtained is large compared to their cost. This allows us to recover the key of KGSnow 2.0 within 2^{294} operations. This is not much more than the exhaustive search of the 256 bits key of Snow 2.0 compared to what we could have expected from an extension of the Billet-Gilbert attack of [5] on the real cipher, or more generally to what we could have expected from an algebraic attack on this type of cipher. It is also more efficient than the classical time-memory trade-off attack $TM = N$. This shows that the key of Snow 2.0 should not be longer than in the current specification.

Acknowledgments. It is clear that the addition modulo 2^n can be described by a system of quadratic equations with extra variables (carries), see [5]. However the idea that this can also be achieved without introducing any extra variable is not trivial and was owe it to Josef Pieprzyk and (independently) Philip Hawkes.

References

1. Armknecht, F., Krause, M.: Constructing Single- and Multi-output Boolean Functions with Maximal Algebraic Immunity. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 180–191. Springer, Heidelberg (2006)
2. Gwenolé, A., Jean-Charles, F.: An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner Bases, INRIA research report, https://hal.ccsd.cnrs.fr/
3. Babbage, S.: A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers European Convention on Security and Detection. In: IEE Conference Publication No. 408 (1995)
4. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proc. International Conference on Polynomial System Solving ICPSS, Paris,France, pp. 71–75
5. Billet, O., Gilbert, H.: Resistance of Snow 2.0 against Algebraic Attacks. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 19–28. Springer, Heidelberg (2005)
6. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In: ASIACRYPT 2000. LNCS, vol. 1975, pp. 1–13. Springer, Heidelberg (2000)
7. Courtois, N.T., Shamir, A., Patarin, J., Klimov, 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)
8. 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)
9. Daemen, J., Rijmen, V.: The Block Cipher Rijndael. In: Schneier, B., Quisquater, J.-J. (eds.) CARDIS 1998. LNCS, vol. 1820, pp. 277–284. Springer, Heidelberg (2000)
10. Diem, C.: The XL-algorithm and a conjecture from commutative algebra. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 323–337. Springer, Heidelberg (2004)
11. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003), http://www.it.lth.se/cryptology/snow/
12. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F_4). Journal of Pure and Applied Algebra 139, 61–88 (1999), www.elsevier.com/locate/jpaa
13. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Workshop on Applications of Commutative Algebra, Catania, Italy, April 3-6, 2002. ACM Press, New York (2002)

14. Fischer, S., Meier, W.: Algebraic Immunity of S-boxes and Augmented Functions. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 366–381. Springer, Heidelberg (2007)
15. Goubin, L.: A Sound Method for Switching between Boolean and Arithmetic Masking. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 3–15. Springer, Heidelberg (2001)
16. Klimov, A., Shamir, A.: A New Class of Invertible Mappings. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 470–483. Springer, Heidelberg (2003)
17. Meier, W., Pasalic, E., Carlet, C.: Algebraic Attacks and Decomposition of Boolean Functions. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 474–491. Springer, Heidelberg (2004)
18. Nyberg, K., Wallén, J.: Improved Linear Distinguishers for SNOW 2.0. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006)
19. Silverman, J.H., Smart, N.P., Vercauteren, F.: An Algebraic Approach to NTRU (q = 2n) via Witt Vectors and Overdetermined Systems of Nonlinear Equations. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 278–293. Springer, Heidelberg (2005)
20. Results of ESTREAM project benchmarks of ESTREAM stream ciphers compared to AES-CTR, RC4 and Snow 2.0, http://www.ecrypt.eu.org/stream/perf/#results
21. MAGMA, High performance software for Algebra, Number Theory, and Geometry, — a large commercial software package, http://magma.maths.usyd.edu.au/

A Proofs of Theorem 1 and 2

Proof (Proof of Theorem 1.). We first prove by induction on $i \geq 1$ that, the carry bit c_i in the addition $x \boxplus y = z$ (see section 2.1 for notations) is equal to 0 with probability $\frac{1}{2} + \frac{1}{2^{i+1}}$. This assertion is true when $i = 0$, as we always have $c_0 = 0$. Let us call p_i the probability that the carry bit $c_i = 0$. We have: $c_i = \lfloor \frac{x_{i-1}+y_{i-1}+c_{i-1}}{2} \rfloor$ (the condition on z does not affect the bits at positions $0 \ldots i-1$ that are random and independent bits). If c_{i-1} is 0, then p_i has a probability $\frac{3}{4}$ to be 0. If c_{i-1} is 1, then p_i has a probability $\frac{1}{4}$ to be 1. Thus we have :

$$p_i = \tfrac{3}{4} \times p_{i-1} + \tfrac{1}{4} \times (1 - p_{i-1}) = \tfrac{1}{2} p_{i-1} + \tfrac{1}{4} \qquad (4)$$

Now if the assertion is true at rank $i-1$, by using equation (4), we compute $p_i = \frac{1}{4} + \frac{1}{2 \cdot 2^i} + \frac{1}{4} = \frac{1}{2} + \frac{1}{2^{i+1}}$, i.e. the assertion is true at level i.

If $i > 0$ and the carry c_i of the addition $x \boxplus y = z$ is 0, and $z_i = 1, ..., z_{i+r} = 1$, we show that we obtain $r+2$ linear I/O equations for $\mathcal{P}$ if $i + r - 1 \leq n - 3$, $r + 1$ linear equations if $i + r - 1 = n - 2$ and r if $i + r - 1 = n - 1$ in the same way as in the proof of Proposition 4.1, namely we simply have $x_i \oplus y_i = 1$ because $c_i = 0$, and then successively we can show that $x_{i+1} \oplus y_{i+1} = 1$ which in turn implies $c_{i+1} = 0$, which in turn gives $x_{i+2} \oplus y_{i+2} = 1$, then $c_{i+2} = 0$ etc. This gives $r + 1$ equations as it goes up to the position $i + r$, where we have $x_{i+r} \oplus y_{i+r} = z_{i+r}$. Moreover, the carry bit $c_{i+r+1} = \lfloor \frac{x_{i+r}+y_{i+r}+c_{i+r}}{2} \rfloor$.

Then if $z_{i+r} = 1$, $c_{i+r+1} = 0$. If $z_{i+r} = 0$, $c_{i+r+1} = \lfloor \frac{2x_{i+r}}{2} \rfloor = x_{i+r}$, and we obtain $z_{i+r+1} = x_{i+r+1} \oplus y_{i+r+1} \oplus x_{i+r}$. In both cases we obtain one more linear equation, that gives a total of $r+2$ linear equations, except when $i+r-1 = n-1$ in which case we get only r, and when $i+r-1 = n-2$ we get $r+1$.

We can observe that when $i = 0$, $p = 1$ and we get the same equations as we always have $c_0 = 0$, in particular when $i = 0$ and $r \geq n-2$ we get again the Proposition 4.1.

To prove the second and third assertion of Theorem 1, we need to observe that $y \boxplus (\bar{y} \boxplus 1) = 0$, where $\bar{y}$ is the result of bitwise complementation of y. Thus we have:

$$x \boxplus y = z \;\Leftrightarrow\; z \boxplus (\bar{y} \boxplus 1) = x \;\Leftrightarrow\; (z \boxplus 1) \boxplus \bar{y} = x \tag{5}$$

In the case of the second assertion, the r least significant bits of z are 0. Then $z \boxplus 1$ becomes $z \oplus 1$ and $c'_1 = \lfloor \frac{\bar{y}_0+1}{2} \rfloor$, that is 0 with probability $\frac{1}{2}$. We obtain by induction that if $c'_1 = 0$, $y_j \oplus x_j \oplus 1 = 0$ for $1 \leq j \leq r-1$ and $y_r \oplus x_r \oplus z_r = 0$. With the least significant bit equation $y_0 \oplus x_0 = 0$ we obtain $r+1$ equations. Another equation comes from the fact that the carry $c_{r+1} = \lfloor \frac{x_r+y_r+c_r}{2} \rfloor$. This implies that if $z_r = 1$, $c_{r+1} = 0$ and if $z_r = 0$, $c_{r+1} = x_r$. As the carry c_{r+1} can be linearly described in both cases, we obtain one more linear equation $z_{r+1} = x_{r+1} \oplus y_{r+1} \oplus c_{r+1}$. We then obtain a total of $r+2$ linear equations, except when $r-1 = n-2$ in which case we get one less, and when $r-1 = n-1$ in case we get two less.

For the third assertion, we consider the carry c''_i in the addition of three numbers: $z \boxplus 1 \boxplus \bar{y} = x$. We assume that $c''_i = 0$ and $z_i = 0, ..., z_{i+r-1} = 0$, we also obtain by induction $r+2$ (or $r+1$ if $i+r-1 = n-2$, or r if $i+r-1 = n-1$) linear I/O equations: $y_j \oplus 1 \oplus x_j = 0$ for $i \leq j \leq i+r-1$, $y_{i+r} \oplus z_{i+r} \oplus x_{i+r} = 0$ and $y_{i+r+1} \oplus z_{i+r+1} \oplus x_{i+r+1} \oplus c_{i+r+1} = 0$ where the carry c_{i+r+1} is zero or x_{i+r}, depending on the value of z_{i+r}.

Here the first carry $c''_1 = \lfloor \frac{z_0+\bar{y}_0+1}{2} \rfloor$ has a probability $\frac{1}{4}$ to be 0. The other carries c''_j can be computed as : $c''_j = \lfloor \frac{z_{j-1}+\bar{y}_{j-1}+c''_{j-1}}{2} \rfloor$. The probability that $c''_i = 0$ for $z \boxplus \bar{y} \boxplus 1 = x$ is computed by induction in the same way as for the first assertion and gives a probability $p''_i = \frac{1}{2} - \frac{1}{2^{i+1}}$.

Proof of Theorem 2. Appears in the extended version of this paper.

B Additional Linear Equations in Attack on KGSnow 2.0

Let us consider the equation (2) (section 6.1) at step 9. According to theorem 2, as $R2_9$ is known, the two least significant bits of $R1_{10}$ can be expressed as linear expressions of the initial state bits. Depending on the value of $R2_9$, more bits of $R1_{10}$ may be expressed in such a way.

In equation (1) at step 10, we can now substitue the two least significant bits of $R1_{10}$ by these linear expressions. Now we observe that in this equation (1), the value of s_{10} is known as it depends on the value of $R1_6$ that has been guessed. Then the value of $z_{10} \oplus R2_{10} \oplus s_{10}$ is know. This implies by theorem 1 that we have two linear boolean equations mixing only initial state bits. Depending on the value of $R2_9$ and $z_{10} \oplus R2_{10} \oplus s_{10}$, we may have more linear expressions.

Towards an Information Theoretic Analysis of Searchable Encryption

Saeed Sedghi, Jeroen Doumen, Pieter Hartel, and Willem Jonker

University of Twente, Enschede, The Netherlands

Abstract. Searchable encryption is a technique that allows a client to store data in encrypted form on a curious server, such that data can be retrieved while leaking a minimal amount of information to the server. Many searchable encryption schemes have been proposed and proved secure in their own computational model. In this paper we propose a generic model for the analysis of searchable encryptions. We then identify the security parameters of searchable encryption schemes and prove information theoretical bounds on the security of the parameters. We argue that perfectly secure searchable encryption schemes cannot be efficient. We classify the seminal schemes in two categories: the schemes that leak information upfront during the storage phase, and schemes that leak some information at every search. This helps designers to choose the right scheme for an application.

1 Introduction

Storage outsourcing is a popular approach towards reducing the total cost of ownership of enterprise data storage. Current solutions either store data in plain, such that the confidentiality of the data is easily compromised, or the data is stored encrypted, which severely limits the kind of service that can be provided. In particular, the ability to search encrypted data is much needed but difficult to provide. Searchable encryption has many applications, particularly where client privacy is a main concern such as in E-mail servers [3], keeping medical information of a client [17], storing private videos and photos, and backup applications [16].

There are two trivial, extreme approaches towards searching in encrypted data. The first trivial approach is for the server to send the client the entire encrypted data base, such that the client may decrypt, then query. Although this solution has a high security, the communication overhead between the server and the client is prohibitively high. The second trivial approach is for the server to decrypt the entire data base, then to execute the query. Although this solution is efficient, letting the server decrypt the data base offers poor security. The problem is thus to find a good compromise between query and data communication efficiency on the one hand, and security on the other hand.

Efficiency means that the query performance should not be influenced negatively by encryption, and that data communication to and from the server should be appropriate. Security means that the stored data, the query that the client

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 345–360, 2008.

sends to the server to retrieve the data selectively, and executing the query on the stored data should not reveal any information to the server about the data except the data items matched with the query.

Three seminal searchable encryption schemes have been proposed [16,3,9], each with specific advantages and disadvantages. A good overview is provided in the PhD thesis of Brinkman [4]. The main problem with each of these proposals is that they assume different limitations: in each case the security analysis assumes a specific model consisting of an adversary with specific limitations. Therefore, the security analysis in each case does not show exactly which parameters reveal information to the server, in a manner that allows us to compare the leakage of schemes.

Contribution. We propose a model for searchable encryption that facilitates an information theoretic security analysis against an adversary with *un*limited power. Since an information theoretic analysis implies no restriction on the computational model of the adversary, information leakage from the parameters is computable in this model. We stress that an information theoretic approach requires idealized encryption and hash functions, and as such this work is an exploration into the theoretical properties of searchable encryption. We apply our model and analysis method to the three seminal approaches towards searchable encryption to show that the model and the analysis method are both general and powerful. The scope of our analysis and results in this paper is limited to a single client using a single server to store a single data item. There can be any number of keywords associated with the data item, and the client may perform any number of keyword searches. The extension to multiple data items, multiple clients and multiple servers is future work.

The paper is organized as follows. Section 2 formalizes the problem. Section 3 presents our approach towards analyzing the security of searchable encryption schemes. Section 4 analyzes the security of the three seminal searchable encryption schemes. A summary of the related work is described in Sect. 5. We present some concrete examples of our analysis in Sect. 6. The last section concludes and suggests future work.

2 Statement of the Problem

Notation. We use the following notation, which is borrowed from Moulin et al [11]. Random variables are denoted by capital letters (e.g. X) and their individual values by lower case letters (e.g. x). The domain over which a random variable is defined is denoted by a script letter (e.g. $\mathcal{X}$) and the number of elements in the range of X is denoted by $|\mathcal{X}|$. The probability mass function (pmf) of a random variable $X \in \mathcal{X}$ is denoted by $P_X(x)$. When no confusion is possible, we drop the subscript to simplify the notation. We write $X \sim P_X$ to indicate that a random variable X is distributed according to P_X. Given random variables X and Y, we denote the entropy of X by $H(X)$, the entropy of X conditioned on Y by $H(X|Y)$ and the mutual information between X and Y by $I(X;Y)$.

2.1 Description of the Problem

There are various formulations of the searchable encryption problem. We propose the following generic formulation in this paper. Without loss of generality we assume that the client splits his data into a non-searchable part d and a set of searchable keywords m. Referring to Fig. 1, let us assume that a client is about to store a tuple $u =< d, m >$ consisting of a single data item d and an associated metadata item m on a server. The metadata m is actually a set of l keywords $m = \{w_1, ..., w_l\}$ where each keyword is taken from a finite set $\mathcal{W}$. The objectives of the client are:

1. The confidentiality of d and m is preserved.
2. d is retrieved in the case m contains a queried keyword.

All solutions to the searchable encryption problem proceed in four phases:

Setup: The client and the server may need to share some data and functions and each may need to prepare some private data.

Storage: The data and the metadata items are transformed to an appropriate format for storage on the server by the steps below:

- The client transforms m to a searchable representation $c = f(m)$, where $f(.)$ is a metadata function.
- The client transforms d to an appropriate encrypted form for storage on the server $s(d, k)$, where $s(., .)$ is a data function and $k \in \mathcal{K}$ is a secret key.
- The client sends the tuple $t =< s(d, k), c >$ to the server for storage.

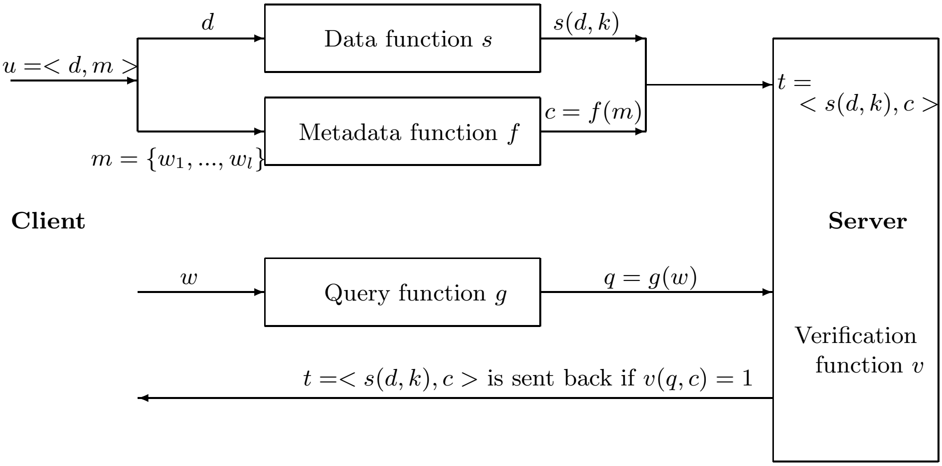


Fig. 1. Formulation of the searchable encryption problem. Here, d is a data item, and $m = \{w_1, ..., w_l\}$ is the associated metadata.

Query: To query the server if a keyword $w \in \mathcal{W}$ occurs in m, the client sends the query $q = g(w)$ to the server, where $g(.)$ is a query function.

Search: Given q and c, using a verification function $v(.,.)$ the server checks if $v(q, c) = 1$, $t =< s(d, k), c >$ is sent back to the client in case of a match.

2.2 Problem Instances

The data d, the metadata m and the functions $s(.,.)$, $f(.)$, $g(.)$ and $v(.,.)$ are the parameters of our proposed searchable encryption formulation. To show that this is a realistic formulation we will instantiate these parameters in such a way that the formulation specializes to the three seminal searchable encryption schemes. These are: Song, Wagner, and Perrig (SWP) [16], Public key encryption with keyword search (PEKS) [3], and Secure indexes (SI) [9]. Below a description of the listed schemes is presented.

The SWP scheme. The first practical approach to the problem of searchable encryption has been proposed by Song, Wagner and Perrig [16]. This scheme does not search for keywords in the metadata; instead searching approaches the data directly. The SWP scheme requires the client to split the data item d into fixed size blocks $d = (b_1, ..., b_l)$ and calculates a searchable representation for each block b_i. However, to apply our formulation to the SWP scheme we consider each block to be a keyword, i.e. $w_i = b_i$.

Setup: The client and the server agree to use a hash function $h_1 : \{0,1\}^v \times \{0,1\}^n \longrightarrow \{0,1\}^{n-v}$, where n is the number of bits in each keyword and $1 \leq v < n$. The client also uses a hash function $h_2 : \mathcal{W} \longrightarrow \{0,1\}^n$.

Storage $f(m)$: To generate a searchable representation $c = f(m)$, the client:

1. Generates a sequence of random values $r_i \in \{0,1\}^v$, $1 \leq i \leq l$.
2. Generates a sequence of trapdoors $x_i = r_i || h_1(r_i, h_2(w_i))$, $1 \leq i \leq l$.
3. Produces $e(w_i, k)$ using a symmetric key encryption function $e(.,.)$, and a secret key $k \in \mathcal{K}$, for each keyword $w_i \in m$.
4. Generates $c_i = x_i \oplus e(w_i, k)$, for each keyword $w_i \in m$.
5. Gathers the sequence $c = (c_1, ..., c_l)$ since the data item is actually a sequence of keywords $d = (w_1, ..., w_l)$.

The client sends the tuple $t =< s(d, k), c >$ to the server for storage, where $s(d, k) = 0$.

Query $g(w)$: To search for a keyword $w \in \mathcal{W}$, the client sends the query $q =< e(w, k), h_2(w) >$ to the server.

Search $v(q, c)$: After receiving q, the server calculates the trapdoor $y'_i || y''_i = c_i \oplus e(w, k)$ for each element c_i of c and checks if $y''_i = h_1(y'_i, h_2(w))$; the server sends back $t =< s(d, k), c >$ to the client in case of a match.

The PEKS scheme. The main disadvantage of the SWP scheme is that the encrypted keyword(s) must be sent to the server for the verification function.

Boneh et al [3] propose the idea of a public key searchable encryption based on the Diffie-Hellman problem. In contrast with the SWP scheme, searching is performed on a set of keywords $m = \{w_1, ..., w_l\}$ associated with the data d.

Setup. The client chooses two groups of prime order p, $\mathbb{G}_1$ and $\mathbb{G}_2$, using group generators g_1 and g_2 respectively and a non-degenerate bilinear function $b : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$. The server and the client agree to use two hash functions $h_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1$ and $h_2 : \mathbb{G}_2 \longrightarrow \{0,1\}^{[log(p)]}$. The client picks a random numbers $\alpha \in \mathbb{Z}_p$ such that $g_1^\alpha \in \mathbb{G}_1$.

Storage $f(m)$: To generate a searchable representation $c = f(m)$, the client:

1. Generates a random value $r_i \in \mathbb{Z}_p$ for each keyword $w_i \in m$.
2. Generates a searchable representation $c_i =< g_1^{r_i}, h_2(b(h_1(w_i), g_1^{(r_i\alpha)})) >$ for each $w_i \in m$.
3. Gathers a set $c = \{c_1, ..., c_l\}$, since the metadata is a set of keywords.

The client sends the tuple $t =< s(d,k), c >$ to the server for storage, where $s(d,k)$ is any appropriate encryption of d.

Query $g(w)$: To query for a keyword $w \in \mathcal{W}$, the client sends $q = h_1(w)^\alpha$ to the server.

Search $v(q,c)$: The server checks for each element c_i of c, if $h_2(b(q, g^{r_i})) = h_2(b(h_1(w_i), g^{(r_i\alpha)}))$; $t =< s(d,k), c >$ is sent back to the client in case of a match.

The SI scheme. Both the SWP and the PEKS schemes have the disadvantage that a search takes time linear in the number of keywords. To obtain a secure, efficient and practical method, Goh [9] proposes an approach to map each keyword to a hash value. The client has a data item d and an associated set of keywords $m = \{w_1, ..., w_l\}$ to store on the server.

Setup. The client chooses $z \geq 1$ independent hash functions $h_1, ..., h_z$, where each $h_i : \mathcal{W} \longrightarrow \{0,1\}^j, j \in \mathbb{N}$.

Storage $f(m)$: To generate a searchable representation $c = f(m)$, the client:

1. Calculates a trapdoor $x_i = \{h_1(w_i), ..., h_z(w_i)\}$ for each keyword $w_i \in m$.
2. Represents c by an array of 2^j bits, where each element c_n, $1 \leq n \leq 2^j$, takes the value 0 or 1 as follows:

$$c_n = \begin{cases} 1 \text{ if there is at least one } h_s(w_i) = n, i = 1, ..., l, s = 1, .., z \\ 0 \text{ Otherwise} \end{cases} \tag{1}$$

The client sends the tuple $t =< s(d,k), c >$ to the server, where $s(d,k)$ is any appropriate encryption of d.

Query $g(w)$: To query for a keyword $w \in \mathcal{W}$ the client sends $q = \{q_1, ..., q_z\}$, where $q_s = h_s(w), s = 1, ..., z$, to the server.

Search $v(q,c)$: The server checks for all $1 \leq s \leq z$ if $c_{q_s} = 1$; $s(d,k)$ is sent back to the client in case of a match.

Table 1 summarizes the functions to be used for $f(.)$, $g(.)$ and $v(.,.)$ by each of the cited methods.

Table 1. Summary of the employed functions in searchable encryption approaches. In this table r is a random number, $h(.)$ is a hash function, $e(.,.)$ is an encryption function, $g(.)$ is a group generator, and $b(.,.)$ is a bilinear map.

	Parameters of the formulation			
Scheme	$t =< d, m >$ $m = \{w_1, ..., w_l\}$	$c = f(m)$	$q = g(w)$	$v(q, c)$
SWP [16]	$d = (b_1, .., b_l)$ $w_i = b_i$	$c_i = e(w_i, k) \oplus x_i$, where $x_i = r_i \|\| h(r_i)$	$q = e(w, k)$	$x'_i \|\| x''_i = c_i \oplus e(w, k)$ Check if $x''_i = h(x'_i)$
SI [9]	d m	$c = \{h_j(w_i)\}$ for $j = 1, ..., z$ and $i = 1, .., l$	$q = \{h_j(w)\}$ for $j = 1, ..., z$	Check if $q \subset c$
PEKS [3]	d m	$c_i =< x_i, y_i >$ where $x_i = g^{r_i}$ and $y_i = h_2(b(h_1(w_i), g^{r_i \alpha}))$	$q = (h_1(w))^{\alpha}$	Check if $h_2(b(q, x_i)) = y_i$

3 Security Evaluation

In this section we evaluate the security of searchable encryption schemes. SWP [16] informally divide the security evaluation of searchable encryption schemes into the following properties:

- Provable security: A searchable encryption scheme is provably secure if the confidentiality of the data and the metadata is preserved before performing any search.
- Hidden query: A searchable encryption scheme is hidden query if a received query does not leak any information on the queried keyword.
- Query isolation: A searchable encryption scheme supports query isolation if the server learns nothing about the metadata after a search is performed.

There are two approaches for evaluating the security of cryptographic schemes: the information theoretic approach and the computational complexity approach. While the information theoretic approach evaluates the security of cryptographic schemes against an adversary with *unlimited* computational power, the computational complexity approach decides whether it is feasible for an adversary with reasonably *limited* computational power to extract information about the plaintext of a ciphertext.

All the previously proposed searchable encryption schemes are proved secure in the computational complexity setting. However, we are interested in the *theoretical* bounds on the security of searchable encryption schemes in general, and the seminal schemes mentioned earlier in particular. Therefore, we use an information theoretic approach to analyze the security of idealized searchable encryption schemes.

In Section 3 we present and motivate four assumptions that formalize the difference between the seminal schemes as published and their idealized interpretations analyzed here.

We now formalize the security evaluation of searchable encryption schemes using the tools of information theory as follows:

- **Provable security:** We formalize the provable security of d as $I(D; s(D, K), f(M))$ (i.e. the information that the stored tuple $< f(M)$,$s(D, K) >$ leaks on D) and the provable security of m as $I(M; s(D, K), f(M))$ (i.e. the information that the stored tuple $< f(M)$,$s(D, K) >$ leaks on M).
- **Hidden query:** We formalize hidden query as $I(W; g(W))$ (i.e. the information that a query leaks on the queried keyword).
- **Query isolation:** We formalize query isolation as: $I(M; v(f(M), g(W)))$ (i.e. the information that a search result leaks on the metadata).

3.1 Security Parameters of the Formulation

In this section we introduce a convenient notation for the information leakage from each searchable encryption parameter. In our formulation we have four different functions that can leak information:

- The stored data $s(d, k)$ leaks on d: $\varepsilon_s = I(D; s(D, K))$.
- The searchable representation $f(m)$ leaks on m: $\varepsilon_f = I(M; f(M))$.
- The query $g(w)$ leaks on w: $\varepsilon_g = I(W; g(W))$.
- The search result $v(f(m), g(w))$ leaks on m: $\varepsilon_v = I(M; v(g(W), f(M)))$.

Here, ε_g and ε_v correspond to hidden query and query isolation respectively. An idealized searchable encryption scheme has perfect security if:

$$\varepsilon_s = \varepsilon_f = \varepsilon_g = \varepsilon_v = 0 \tag{2}$$

Theorem 1, relates provable security to ε_s and ε_f by giving upper bounds on the uncertainty of the server about the data and the metadata items, under the condition that the server has access to $s(d, k)$ and $f(m)$.

Theorem 1. *The provable security of any searchable encryption scheme admits the following upper bound:*

$$I(D; s(D, K), f(M)) \leq \varepsilon_s + \varepsilon_f - H(M|D) + H(M|D, f(M))$$
$$I(M; s(D, K), f(M)) \leq \varepsilon_f + \varepsilon_s - H(D|M) + H(D|M, s(D, K))$$

Proof: see the technical report [14].
The intuition of the first inequality is as follows: ε_s is the information that the stored data $s(D, K)$ leaks on D directly, and $\varepsilon_f - H(M|D) + H(M|D, f(M))$ is the information that the searchable representation $f(M)$ leaks on D indirectly via M. A similar intuition applies to the second inequality.

4 Analysis of Known Schemes

The security of searchable encryption schemes is analyzed in related work using the computational complexity approach. The reason is that the cryptographic primitives which are used for the data, the metadata, the query and the verification function (e.g. block ciphers and hash functions) are not information theoretically secure. Here, we are interested in an adversary with unlimited power, who however cannot look inside the cryptographic primitives. In other words, we analyze whether searchable encryption schemes leak information to the server under the assumption that perfectly secure cryptographic primitives are used:

Assumption 1. Any encryption function is a one-time pad encryption since only the one time pad encryption has been proved to be information theoretically secure.

Assumption 2. The client uses a secret table by way of a hash function. In most cases considered here, the server does not need to know the hash function h. A secret table works as follows: given $y = h(x)$, and a string y it is *impossible* to find the corresponding bit string x, whereas in the case that $h(.)$ is a hash function, it is *hard* to find the string x. Therefore, a secret table is information theoretically secure and $I(W; h(W)) = 0$.

We make the following assumptions on the distribution of the keywords in the metadata:

Assumption 3. The distribution of the total number of keywords in m is $L \backsim P_L(l)$. i.e. clients tend to choose the number of keywords according to a distribution which depends on the application domain.

Assumption 4. Given the total number of keywords, l, the distribution of choosing a keyword $P(w \in \mathcal{M} | L = l)$ is uniform. i.e. the client picks each keyword from the set $\mathcal{W}$ with the same probability. Although this assumption might not be correct for all practical situations, the uniform distribution of keywords gives the highest security in comparison with the other distributions.

These assumptions are purely theoretical. In particular, we are not suggesting to use a one time pad encryption as in this case outsourcing data to the data base would not help the client (i.e. the secret key must be as large as the data itself). The hash table is just as impractical. However, these assumptions allow us to analyze how close the seminal schemes are to ideal security. In other words, by these assumptions the client uses the most secure cryptographic primitives (for the data, metadata, query and verification functions) and keyword distribution, and we explore the theoretical upper bounds on the security of these schemes. Moreover, our results show how much information the parameters of the searchable encryption schemes leak to the server.

When it comes to a practical implementation, the leakage of information from the functions can not be smaller than what is derived here. The reason is that although the cryptographic primitives are secure against an adversary with limited power, they are not information theoretically secure. Therefore, our results show:

1. The minimum leakage of information from the functions of searchable encryption in all situations.
2. Which functions of a searchable encryption scheme leak more information, and which functions leak less information. This type of analysis is not possible in computational security settings.

4.1 Idealized SWP

The parameters of the SWP scheme are as follows: the data and the metadata are the same $d = m$, there is no stored data on the server $s(d, k) = 0$, the metadata is a sequence of keywords $m = (w_1, ..., w_l)$, for each keyword $w_i \in m$ a random value r_i is generated. According to assumption 1, the encryption function $e(.,.)$ is a one time-pad encryption. Hence, each keyword $w_i \in m$ is encrypted using a unique key $k_i \in \{0, 1\}^n$ as $e(w_i, k_i)$.

– **The information leakage from** $s(D, K)$: Since there is no stored data in the SWP scheme $(s(D, K) = 0)$,

$$\varepsilon_s = I(D; s(D, K)) = 0 \tag{3}$$

– **The information leakage from** $f(M)$: Let define the function

$$T(e(m, k)) = e(m, k) \oplus (r_1||h(r_1), ..., r_l||h(r_l)) \tag{4}$$

then $f(m) = T(e(m, k))$, where $k = (k_1, ..., k_l)$. Since $M \longrightarrow e(M, K) \longrightarrow T(e(M, K))$ forms a Markov chain:

$$I(M; T(e(M, K))) \leq I(M|e(M, K)) \tag{5}$$

Using one time pad encryption for $e(m, k)$, $I(M|e(M, K)) = 0$ [15]. Hence,

$$\varepsilon_f = I(M; f(M)) = 0 \tag{6}$$

Therefore, the searchable representation achieves perfect secrecy.

– **The information leakage from** $g(W)$: To query for a keyword w the client sends the query $g(w) =< e(w, k), h_1(w) >$ to the server. Hence,

$$\varepsilon_g = I(W; e(W, K), h_1(W)) \tag{7}$$

Using standard information theoretic formulas:

$$\varepsilon_g = I(W; h_1(W)) + I(W; e(W, K)|h_1(W)) \tag{8}$$

According to assumption 2, $I(W; h(W)) = 0$, and according to assumption 1, $I(W; e(W, K)) = 0$. Hence, $\varepsilon_g = 0$. Therefore, a query does not leak any information about the keyword queried.

- **The information leakage from** $v(f(M), g(W))$: The SWP scheme transforms each keyword w_i to a unique searchable representation c_i, since each keyword is transformed to a searchable representation by a unique random value r_i. Therefore, if the keywords w_i and w_j are the same keywords in the metadata, c_i and c_j are different. However, if the client sends the query $g(w_i)$ (or $g(w_j)$) to the server, by the search result the server learns that c_i and c_j are searchable representations of the same keywords. Therefore, the uncertainty of the server about c_i and c_j reduces to the uncertainty about c_i (or c_j) alone. The following theorem quantifies the information that a search leaks on M.

  Theorem 2. *Let* $P(M|L) = \frac{1}{|\mathcal{W}|^l}$*,* $\mathcal{E}(S)$ *denotes the expected number of repeated keywords and* $\mathcal{E}(L)$ *denotes the expected number of keywords in* M*. Then,*

  $$\varepsilon_v = I(M; v(f(M), g(W))) = H(W)(\mathcal{E}(S) - 1) \tag{9}$$

  where $\mathcal{E}(S) \approx (\frac{1}{|\mathcal{W}|})$*.*

  Proof: see the technical report [14].
  Intuitively, Theorem 2 says that the uncertainty of the repeated keywords is the reductions in the uncertainty of M due to the knowledge of a search result. If each keyword occurs once in M, then $\mathcal{E}(S) = 1$ and $I(M; v(f(M), g(W))) = 0$, and if all the keywords are the same, then $\mathcal{E}(S)) = \mathcal{E}(L)$ and $I(M; v(f(M), g(W))) = \mathcal{E}(L) - 1$.

Summarizing, even using perfectly secure cryptographic primitives for the SWP scheme, a search reveals some information to the server. However, storing data does not leak any information.

4.2 Idealized SI

Without loss of generality we assume that the client uses only one hash function $h(.)$ to map each keyword w to a searchable representation $h(w)$. (i.e. $f(m) = \{h(w_1), ..., h(w_l)\}$). According to assumption 2, the client uses a secret table by way of the hash function. We also assume that $P(M|L) = \frac{1}{\binom{|\mathcal{W}|}{l}}$, since according to assumption 4 the distribution of keywords in the metadata is uniform and the metadata is a set of keywords.

- **The information leakage from** $s(D, K)$**:** Since we assume that encryption is one time-pad encryption (assumption 1), the stored data achieves perfect secrecy [15]. Hence,

  $$\varepsilon_s = I(D; s(D, K)) = 0 \tag{10}$$

- **The information leakage from** $f(M)$**:** In contrast with the SWP scheme, the SI scheme admits false positives. A false positive occurs when two or more different keywords are mapped to the same searchable representation by $h(.)$. First we evaluate the information leakage in the case without false positives. We then extend the evaluation in the case of a probability of a false positive.

- Without false positives

$$\varepsilon_f = I(M; f(M)) = I(M; h(W_1), ..., h(W_L)) \tag{11}$$

Theorem 3 shows the information leakage from $f(M)$ in the case without false positive.

Theorem 3. *The information that the searchable representation leaks on the metadata is:*

$$\varepsilon_f = I(M; f(M)) = H(L) \tag{12}$$

where $H(L)$ is the uncertainty in the number of unique keywords in the metadata.

Proof: See the technical report [14].
Intuitively, Theorem 4 says that the uncertainty in the number of unique keywords contained in the metadata is the reduction in the uncertainty of the metadata due to the knowledge of the searchable representation.

- A False positive
In this case the server cannot learn the precise number of unique keywords since more than one unique keywords could be mapped to the same searchable representation. The following theorem shows how the probability a false positive P reduces the information that $f(M)$ leaks on M.

Theorem 4. *Let j be the number of bits in the output of the hash values. Then,*

$$\varepsilon_f = I(M, f(M)) = H(L) - \beta \tag{13}$$

Here, $\beta = \sum_{i=1}^{|\mathcal{W}|} \sum_{x=1}^{i} (P(i) log(\frac{1}{\binom{\mathcal{W}}{i}}) + \frac{(2^j)! x^{i-l}}{(2^j - x)! 2^{ji}} P(i) log(\frac{(2^j)! x^{i-x}}{(2^j - x)! 2^{ji} \binom{\mathcal{W}}{i}}))$.

Proof: see the technical report [14].
Intuitively, Theorem 4 says that a false positive reduces the revealed information about the uncertainty of the number of keywords to the server.

- **The information leakage from $g(W)$:** To query for a keyword w the client sends the query $g(w) = h(w)$ to the server. According to assumption 2:

$$\varepsilon_g = I(W; h(W)) = 0 \tag{14}$$

Therefore, a query in the SI scheme does not reveal any information about the queried keyword.

- **The information leakage from $v(f(M), g(W))$:** In contrast with the SWP scheme, the metadata is a set of keywords. Hence, each keyword in the metadata is unique and by a search result the server learns nothing about the metadata.

$$\varepsilon_v = I(M; v(f(M), g(W))) = 0 \tag{15}$$

Summarizing, using even perfectly secure cryptographic primitives for the SI scheme, the searchable representation reveals information to the server. The probability of false positive increases the security of the SI.

4.3 Idealized PEKS

The PEKS scheme was originally proposed for transforming a *set* of keywords to a searchable representation. However, the scheme is capable of transforming a *sequence* of keywords to a searchable representation as well, since a different random value r_i is considered to transform each keyword w_i to a searchable representation (see section 2.4 PEKS). In other words, the same keywords can be mapped to a different searchable representation. Therefore we consider the metadata as a sequence of keywords. Since an information theoretic approach cannot handle public key cryptography, we analyze only the hash function of the PEKS scheme. In other words, our analysis relies on the security of the hash function only.

- **The information leakage from** $s(D,K)$**:** According to assumption 1, $s(D,K)$ is one time-pad encryption. Hence,

$$\varepsilon_s = I(D, s(D,K)) = 0 \tag{16}$$

  Therefore, the stored data does not reveal any information about the data.

- **The information leakage from** $f(M)$**:** Since the client uses a secure table and a unique random value to transform each keyword to a searchable representation, the server cannot learn anything about the metadata. Hence,

$$\varepsilon_f = I(M; f(M)) = 0 \tag{17}$$

  Therefore, the server cannot learn anything about the metadata due to the knowledge of the searchable representation.

- **The information leakage from** $g(W)$**:** To query for a keyword w the server sends the query $g(w) = h(w)^\alpha$ to the server. According to assumption 2,

$$\varepsilon_g = I(W; h(W)^\alpha) = 0 \tag{18}$$

  Therefore, the server cannot learn anything about the plaintext of the query after receiving a query.

- **The information leakage from** $v(f(M), g(W))$**:** Similar to the case of the SWP scheme, before performing any search by the client, the server cannot learn the unique keywords in the metadata. However, after a search for the keyword w, the server learns the repeated keywords in case of a match. By the same analysis as of Theorem 2, the information leakage from a search is calculated as follows:

$$\varepsilon_v = I(M; v(f(M), g(W))) = H(W)(\mathcal{E}(S) - 1) \tag{19}$$

Table 2. Summary of the information leakage from the parameters of the SWP, SI, and PEKS schemes, as well as the two trivial schemes. In this table $\beta = \sum_{i=1}^{|\mathcal{W}|} \sum_{x=1}^{i} (P(i) log(\frac{1}{\binom{\mathcal{W}}{i}}) + \frac{(2^j)!x^{i-l}}{(2^j-x)!2^{ji}} P(i) log(\frac{(2^j)!x^{i-x}}{(2^j-x)!2^{ji}\binom{\mathcal{W}}{i}}))$ (see 4.2).

	Information leakage from			
Scheme	Stored data ε_s	Searchable representation ε_f	Query ε_g	Search ε_v
Schemes that do not leak at all				
Trivial solution 1	0	0	0	0
Schemes that leak at each search				
Trivial solution 2	0	0	$H(W)$	$H(D)$
SWP	0	0	0	$(\mathcal{E}(S) - 1)H(W)$
PEKS	0	0	0	$(\mathcal{E}(S) - 1)H(W)$
schemes that only leak up front				
SI	0	$H(L) - \beta$	0	0

Summarizing, using even perfectly secure cryptographic primitives for the PEKS scheme, a search reveals some information to the server.

Table 2 summarizes the information leakage from the parameters of the SWP, SI and PEKS schemes. In this table the analysis of the two trivial schemes from the introduction is included for comparison. Trivial solution 1 lets the server return the entire encrypted data base to the client at each query. In this case the server never learns anything about the data, hence $\varepsilon_s = \varepsilon_f = \varepsilon_g = \varepsilon_v = 0$. Trivial solution 2 lets the server decrypt all data at the first query, hence, the uncertainty in the keywords $H(W)$ and the data $H(D)$ is leaked to the server at the first query.

We conclude by categorizing the seminal searchable encryption schemes into three groups: (1) schemes that do not leak at all, (2) schemes that leak everything up front, and (3) schemes that leak nothing up front but which leak at each search. In practice this means that if an application is expected to perform many searches, a scheme from the second group is probably best. If we have a scenario where only a few searches are expected, a scheme from the third group is best.

5 Related Work

Our analysis of the information leakage of searchable encryption is limited to the three seminal schemes, SWP, PEKS and SI. However, we believe that the same analysis can be applied to more recent schemes based on the seminal schemes. Here, we discuss the most prominent searchable encryption schemes that follow the seminal schemes.

SWP based schemes. For XML documents Brinkman et al [5] modify the SWP scheme to facilitate a faster search by exploiting the tree structure of XML documents.

SI based schemes. Chang and Mitzenmacher [6] propose a scheme based on mapping keywords of the metadata to hash values. The proposed scheme has a lower probability of false positives than the SI scheme and the scheme also hides the number of keywords in the metadata from the server. Curtmola et al [7] propose a symmetric key encryption approach that offers better efficiency than the SI scheme, and the scheme can be applied for multiple users scenarios by generating a random value for each user.

PEKS based schemes. The schemes proposed by Abdalla et al [10] lowers the probability of false positives in the PEKS scheme. Bellare et al [2] propose a deterministic searchable encryption scheme that offers faster search than the PEKS scheme by mapping each searchable representation to an index. Baek et al [1] propose a modified PEKS in such a way that the client could send a query through an insecure channel by mapping each keyword to several hash values. Park et al [13] propose a modified PEKS that gives a proxy the ability to decrypt searchable representations containing desired keywords. Fuhr at al [8] propose a scheme that allows the client to recover the plaintext of the keywords after transforming the metadata to a searchable encryption. The scheme applies the xor operation to the searchable representation and the hash value of a random value. The random value is kept by the client.

None of the extensions to the seminal schemes use primitives that cannot be idealized in the same way as we have idealized encryption and hashing. Therefore we believe that our methods are applicable to the extensions of the seminal schemes. To prove this is future work.

6 Example

In this section we present a few numerical examples to illustrate what the analysis actually means. Our cryptographic primitives in this example are chosen according to assumptions 1 and 2. According to assumption 4, the distribution of the keywords in the metadata is uniform and we consider a Poisson distribution for the distribution of the number of keywords ($P(l)$) in the metadata as follows [12]:

$$P(l) = \frac{\lambda^l e^{-\lambda}}{l!} \tag{20}$$

Here λ is the expected number of keywords in the metadata. Let the cardinality of keyword in metadata be equal to the size of Oxford dictionary, 126000 words. Moreover, let the expected number of keywords in the metadata be $\lambda = 100$.

SWP and PEKS: The expected number of the repeated keywords $\mathcal{E}(S) = 16.9\frac{1}{126000}$ and the enctropy of keywords is $H(W) = 16.9$ Hence the information leakage from the parameters is then as follows, $\varepsilon_d = \varepsilon_f + \varepsilon_g = 0$, and $\varepsilon_v = 16.9\frac{1}{126000}$.

SI: The SI scheme is applied to a set of keywords. Let assume there is no false positives. The entropy of the number of the number of keywords is $H(L) = 5.14$.

Hence, the information leakage from the parameter is: $\varepsilon_d = \varepsilon_g = \varepsilon_v = 0$ and $\varepsilon_f = 5.14$. Now, let the number of in the hash of keywords be $j = 200$, Then, $H(L) = 4.7$.

The example shows that in the case that the client performs only a few searches on the stored data (e.g. backup scenarios), the SWP scheme or the PEKS scheme would be ideal. However, In the case that the client intends to send many queries to the server, the SI scheme is a better choice since this scheme has higher efficiency.

7 Conclusion and Future Work

We present an information theoretic analysis of searchable encryption, where a single client stores a single data item on a single server. Any number of keywords may be associated with the data item and the client can perform any number of queries. We propose a generic formulation for the searchable encryption problem and apply the formulation to all three seminal schemes (SWP, SI, and PEKS) from the literature. We then formalize the security of searchable encryption using the tools of information theory and analyze the seminal schemes. The results of our analysis shows that even using perfectly secure cryptographic primitives, the parameters of all seminal schemes leak some information to the server. Our analysis shows that each scheme has its specific strengths and weaknesses in terms of provable security, hidden query and query isolation, and thus provides a useful tool to compare searchable encryption schemes. In future work we intend to extend the analysis to multiple data, multiple client and multiple server settings.

Acknowledgements

We thank Henk van Tilborg, Qiang Tang and Peter van Liesdonk for their help with the paper.

This research is supported by the SEDAN project, funded by the Sentinels program of the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs under project number EIT.7630.

References

1. Baek, J., Safiavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. Available on Cryptology ePrint Archive, Report 2005/119 (2005)
2. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
3. 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, pp. 506–522. Springer, Heidelberg (2004)

4. Brinkman, R.: Searching in encrypted data. PhD thesis, University of Twente, Enschede, The Netherlands (2007), ISBN 9789036524889
5. Brinkman, R., Feng, L., Doumen, J.M., Hartel, P.H., Jonker, W.: Efficient tree search in encrypted data. Information Systems Security Journal 13(3), 14–21 (2004)
6. Chang, Y.C., Mitzenmacher, M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531. Springer, Heidelberg (2005)
7. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS 2006: Proceedings of the 13th ACM conference on Computer and communications security, pp. 79–88. ACM, New York (2006)
8. Fuhr, T., Paillier, P.: Decryptable searchable encryption. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 228–236. Springer, Heidelberg (2007)
9. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003)
10. Malone-Lee, J., Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
11. Moulin, P., O'Sullivan, J.A.: Information-Theoretic analysis of information hiding. IEEE Transactions on information theory 49(3), 563–593 (2003)
12. Papoulis, A.: Probability, Random Variables, and Stochastic Processes, 4th edn (2002)
13. Park, D., Cha, J., Lee, P.: Searchable keyword-based encryption. Cryptology ePrint Archive, Report 2005/367 (2005), http://eprint.iacr.org
14. Sedghi, S., Doumen, J.M., Hartel, P.H., Jonker, W.: Towards an information theoretic analysis of searchable encryption (extended version). Number TR-CTIT-08-50, Enschede, August 2008. University of Twente, http://eprints.eemcs.utwente.nl/13176/
15. Shannon, C.: Communication theory of secrecy systems. Bell Sys. Tech. 28(4), 656–715 (1949)
16. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 21st Symp. on Security and Privacy (S&P), Berkeley, California, May 2000, pp. 44–55. IEEE Computer Society, Los Alamitos (2000)
17. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient DNA searching through oblivious automata. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 519–528. ACM, New York (2007)

A Bootstrap Attack on Digital Watermarks in the Frequency Domain

Sam Behseta[1], Charles Lam[2], and Robert L. Webb[3]

[1] Department of Mathematics, California State University,
Fullerton, CA, 92834, USA
statistician@gmail.com

[2] Department of Mathematics, California State University,
Bakersfield, CA, 93311, USA
clam@csub.edu

[3] Department of Computer Science, California Polytechnic State University,
San Luis Obispo, CA, 93407, USA
webb@calpoly.edu

Abstract. In this paper, we propose five simple algorithms to execute a collusion attack given several watermarked documents. Each document considered is a picture represented as a matrix of two dimensional Discrete Cosine Transform (DCT2) coefficients. Our algorithm is independent of media type. Bootstrap methods are used to construct confidence intervals for each DCT2 coefficient and determine its uncertainty. Using simulation studies we show that Bootstrap procedures are highly efficient with respect to the number of iterations and sample size per iteration while maintaining stellar probabilistic coverage, providing results at least as good as averaging or taking the median of signals. Most importantly, a set of simulation studies suggest that the precision of our heuristic methodology increases quickly when the number of watermarked copies are increased, but good probabilistic coverage is achieved with a low number of independently watermarked copies. We conjecture that the Bootstrap methodology will be highly effective in reconstructing the original signal for documents with high redundancy.

1 Introduction

A secure spread spectrum digital watermark is a signal added into a digital document in such a way that the alteration to the final use, usually analog, of the digital signal is minimal. However, the watermark is strategically placed so that common operations on the signal will not destroy it. Since the watermark can be recovered from the host signal if the original signal is available, the watermark can be used to identify the original owner of a specific copy [3]. For a discussion of the overall motivations for digital watermarking, see [3,15].

The questions of security and robustness of these types of watermarks have stimulated a large number of research activities. For example, see [3,6,10,11,19,25]. In this paper, we describe resampling algorithms to determine a confidence interval for the true value of any part of the original signal in a

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 361–375, 2008.

collusion attack. This coverage shows that we can obtain knowledgeable information about the original unwatermarked document, when only a small number of watermarked documents are available. By using a measure of centrality such as the median, we can also produce a document at least as accurate to the original document as averaging signals of all available watermarked documents.

Our algorithms generate two statistics that can be used to determine the original signal: the percentage of altered 2D Discrete Cosine Transform (DCT2) coefficients that are inside a particular confidence interval, and the average span of that confidence interval. We develop methods which can be used to make confidence intervals as narrow as possible without effecting their coverage rate.

The creators of secure spread spectrum digital watermarks attempted to approximate the original signal by averaging all of the available watermarked signals [3]. This method is satisfying because each watermarked signal is created by adding a sequence with low correlation.

Unfortunately, simple averaging does not satisfactorily reconstruct the original image unless the number of attackers is large. The metric to determine the similarity of watermark W with some other watermark W^* is

$$\mathrm{sim}(W, W^*) = \frac{W \cdot W^*}{\sqrt{W \cdot W}}. \tag{1}$$

Let $W_1, W_2, ...W_n$ be n sequences with low correlation. Let $W^* = \frac{W_1+W_2+...+W_n}{n}$, then

$$\mathrm{sim}(W_1, W^*) = \frac{W_1 \cdot \left(\frac{W_1+W_2+...+W_n}{n}\right)}{\sqrt{W_1 \cdot W_1}} \approx \frac{W_1 \cdot W_1}{n\sqrt{W_1 \cdot W_1}} \approx \frac{\mathrm{sim}(W_1, W_1)}{n}.$$

When sequences follow the white noise assumption, then $\mathrm{sim}(W_1, W_1) \approx \sqrt{L}$, where L is the length of the sequence W_1. Therefore, if $\frac{\sqrt{L}}{n}$ is large enough to register a positive match between two watermarks, then simple averaging is insufficient to obscure the original watermark.

If we continue to assume that watermarks are drawn from $N(0, 1)$, and if $\mathrm{sim}(W_1, W^*) = s$, then the probability that W_1 and W^* are unrelated is the same as the probability of choosing a sample from $N(0, 1)$, which is s standard deviations away from the mean. This is demonstrated by the fact that if $W = w_1, w_2, ..., w_L$ is a fixed sequence of numbers from $N(0, 1)$, and $W^* = w_1^*, w_2^*, ..., w_L^*$ is independently generated from samples of $N(0, 1)$, then

$$W \cdot W^* = w_1 w_1^* + w_2 w_2^* + ...w_L w_L^*$$

will follow the distribution $N(0, w_1^2 + w_2^2 + ... + w_L^2)$. Notice that the standard deviation of this distribution is $\sqrt{W \cdot W}$. Therefore, $\mathrm{sim}(W, W^*)$ will follow the $N(0, 1)$ distribution if W^* is independently generated from a fixed W. For the averaging collusion attack to be effective, n must be very large or L must be very small. However, these are generally unreasonable assumptions [3]. The algorithms in this paper address these issues by using a small number of attackers to find a narrow confidence interval with a high probability of containing the true value of the original signal. We also show that choosing the median of the generated confidence interval produces a result at least as good as the average.

Our work was initially inspired by Cox, et al. [3]. They detailed a watermarking system that is used to determine the original owner of a signal. In this paper, we use the watermarking method described by Zhu, et al. [25], which is a modification of the method presented by Cox [3]. The term "digital watermark" has been used for many other purposes (see Furon [11], Delaigle, et al. [4], Cannons and Moulin [1], Soriano, et al. [17], Wolfgang, Podilchuk and Delp [21] and references therein).

Collusion attacks have been specifically addressed by Doërr and Dugelay [6] where redundancy in the host signal was used to distort the included watermark. Vinod and Bora [19] used similar methods on video signals, where redundancy is especially applicable. These types of geometric attacks can be mitigated using the methods presented in Dong [5].

The specific problem of this paper has been considered by Comesana, et al. [2] Their paper also includes an especially detailed discussion of the difference between the motivations for watermarking presented in [18] and [3]. A theoretical result on the limitations of the method presented in [3] and a possible attack is presented in Ergun, et al [10]. However, there are perceptible negative effects on the host signal when this attack is used.

More recent work on collusion attacks have been done in [23,24,22,20]. In this series of papers, the authors classify collusion attacks into linear and non-linear categories where simple averaging is the most naive of the linear attacks and using the median of the colluding copies is the most naive of the non-linear attacks. Specifically, in [24], the authors introduce a non-linear attack that reduces the similarity metric. Further work on non-linear attacks has been done by [14,13]. The attack presented in our paper is non-linear, but extends previous schemes by using the non-parametric Bootstrap methodology to mitigate possible effects of outliers in the parametric watermark.

The term "watermark" has been used in the literature for other purposes. Included are references to papers explaining these other uses. For a discussion of video watermarking see Podilchuk and Delp [15]. The relationship between video and image watermarking is explored in Doerr and Dugelay [7]. Another application for watermarking is steganography where a detailed signal is hidden in a host signal. Vila-Forcén, et al [18] have used collusion attacks to extract information about the host signal and original signal in this setting.

This paper is organized in the following way: We describe our notations and algorithms in section 2. In section 3, we summarize the extensive empirical simulations we performed using the "Lena" picture as the host signal. The results demonstrate that the output of the algorithms are effective in terms of speed and ability to determine precise confidence intervals with high probability of containing the true value of the original signal. At the end of section 3, we include a discussion of the technical details of the simulation. Section 4 contains a discussion of our methods and also includes some suggestions on future work. Tables containing the numerical results are presented in the appendix.

2 Methods

2.1 The Bootstrap Methodology

Let $X_1^*, ...X_n^*$ be an independently drawn and identically distributed sample from $\hat{F}$, the empirical cumulative distribution function of a dataset $X_1, ..., X_n$. $X_1^*, ...X_n^*$, or the Bootstrap sample, can be acquired by sampling with replacement from $X_1, ..., X_n$. Also, suppose that T_n is an estimator of some unknown parameter θ (for example, $T_n = \bar{X}$). The Bootstrap sample is nonparametric in a sense that since it is obtained from the dataset, it makes no assumptions regarding the underlying statistical model and its parameters. By generating repeated Bootstrap samples, one can obtain a probability distribution for T_n^*, hence being able to assess its uncertainty.

Let $U_F(x) = Pr(T_n \leq x)$ denote the distribution of T_n. By replacing F with $\hat{F}$, we have $U_{\text{boot}}(x) = U_{\hat{F}}(x) = Pr[T_n^* \leq x|(X_1, ..., X_n)]$, the conditional distribution of T_n^* given data. Let ρ_∞ be a metric generated by a sup-norm[1] on the space of all distributions in R^p, for an integer p, representing the dimensionality of the distribution space. Then the following results hold (See Shao and Tu [16] for details):

(1) If T is continuously ρ_∞-Hadamard differentiable, then the Bootstrap estimator U_{boot} is strongly consistent. In other words, $\rho_\infty(U_{\text{boot}}, U_n) \to 0$ (almost surely).

(2) If T is continuously ρ_r-Frechet differentiable, then the Bootstrap estimator U_{boot} is strongly consistent.

These results guarantee the fact that the Bootstrap distribution U_{boot} is consistent for many estimators such as the mean and the median.

2.2 Notation

Let the original signal be represented by an $I \times J$ matrix D_0 such that the value of the (i, j)-th element is $D_{0,i,j}$, where $i = 1, ..., I$, $j = 1, ..., J$. In this context, each element is the value of a DCT2 coefficient. There are also K independently watermarked documents which will be used to make statistical inference with regard to the original signal. These matrices are denoted by D_k, where $k = 1, ..., K$. Similarly, $D_{k,i,j}$ represents the (i, j)-th element of the k-th matrix, also a DCT2 coefficient.

The Bootstrap is performed by sampling from K existing signals. See [8] for a discussion of the theoretical properties of the Bootstrap procedure. The probabilistic framework for determining a confidence interval for $D_{0,i,j}$ is obtained by sampling n many elements with replacement from $D_{1,i,j}, \ldots, D_{K,i,j}$, where $n < K$ followed by calculating some measure of centrality from this sample. This procedure is repeated B times to create B possible values of the given statistic.

[1] $\parallel h \parallel_\infty$ is the sup-norm of a function h on R^p, if $\parallel h \parallel_\infty = \sup_x |h(x)|$.

For our simulation, the K signals are created by adding samples taken from $N(0,1)$ to some of the elements of the original signal. For some subset $S \subset \{(i,j) : i = 1, ..., I, j = 1, ..., J\}$, we let

$$D_{k,i,j} = D_{0,i,j} + \alpha W_{k,i,j}, (i,j) \in S, \tag{2}$$

for some appropriate constant α, where $W_{k,i,j}$ is a sequence of numbers from $N(0,1)$. The constant α can be viewed as the standard deviation of the added noise.

Using a simulation technique described in the next section, we calculate confidence intervals to determine the average span of each interval and the empirical percentage of the times that the original signal is contained in the interval.

2.3 Algorithms

We propose five simple algorithms to calculate confidence intervals for the original signal. There are three parameters shared by all proposed algorithms: the number of watermarked signals (K), the number of Bootstrap iterations (B), and the number of samples per iteration (n), hereafter referred to as "sample size". The basic framework of all algorithms is described in Algorithm 0.

Let (i,j) be a typical element of the signal. There are K associated values of this element, one in each of $D_{k,i,j}$, where $k = 1, ..., k$. In this paper, we sample n of the elements with replacement. Next, we calculate two types of measurements. First, a metric that reflects the distance to a pre-assigned measure of centrality. Second a central statistic such as the mean. Finally, this process is repeated B times to generate B bootstrapped observations. Let $C_{i,j}$ be the list of B values in increasing order. Let $C_{i,j}^{(p)}, p \in (0,1)$ be the value in $C_{i,j}$ so that $[pB]$ of the values are smaller than $C_{i,j}^{(p)}$. The median of those B values ($C_{i,j}^{(0.5)}$) is the bootstrapped median.

Additionally, we calculate associated confidence intervals. Let $\gamma \in (0,1)$. The Bootstrap $(1-\gamma)$-confidence interval for (i,j) is defined as

$$(C_{i,j}^{(\gamma/2)}, C_{i,j}^{(1-\gamma/2)}). \tag{3}$$

The average confidence span is defined as

$$\frac{\sum_{(i,j)\in S}\left(C_{i,j}^{(1-\gamma/2)} - C_{i,j}^{(\gamma/2)}\right)}{|S|\alpha}, \tag{4}$$

where S is the subset of the signal that has been modified by the watermarking process. The appearance of α in the denominator is motivated by the fact that, in the initial watermark process, the signals were magnified by a scale of α. Since we know the original signal, we can also calculate the percentage of the confidence intervals that actually capture $D_{0,i,j}$, the value of the original signal. For any $(1-\gamma)$-confidence interval, this can formally be stated as

$$\frac{\sum_{(i,j)\in S} V_{i,j}^{(1-\gamma)}}{|S|}, \tag{5}$$

where

$$V_{i,j}^{(1-\gamma)} = \begin{cases} 1 & \text{when } D_{i,j} \in (C_{i,j}^{(1-\gamma/2)}, C_{i,j}^{(\gamma/2)}) \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 0: The General Approach

(0.1) For the element (i,j), sample with replacement n many observations with $n < K$.

(0.2) Define a Metric M which can be implemented pointwise on each of the IJ elements. Thus, the metric can be presented as $M_{i,j}$, for $i = 1, ..., I$ and $j = 1, ..., J$.

(0.3) Calculate a statistic $R_{i,j}$ based on the implemented metric on the set of n sampled elements.

(0.4) Repeat steps (2), (3), and (4), B many times.

(0.5) Consider an appropriate quantile or an appropriate measure of centrality of the probability distribution of $R_{i,j}^B$, along with a variability measure associated with it. The superscript B here is to emphasize the number of the Bootstrap repetitions.

(0.6) Form a pointwise $1-\gamma$ percentile level confidence interval for the statistic. In other words, form a confidence interval for the element (i,j) with $C_{i,j}^{(\gamma/2)}$, and $C_{i,j}^{(1-\gamma/2)}$ as its lower and upper bounds respectively.

(0.7) Calculate the pointwise width (span) of the confidence interval of step (7).

(0.8) Calculate the percentage of the confidence intervals covering the actual element of the signal D'.

Algorithm 1: Detection Via Boostrapping the Mean

(1.3) Take $R_{i,j} = \frac{\sum_l D_{l,i,j}}{n}$, where l represents the bootstrapped sample.

(1.5) Consider Median$(R_{i,j}^B)$ as the measure of centrality.

Algorithm 2: Detection Via Boostrapping the Median

(2.3) Take $R_{i,j} = \text{Median}_l(D_{l,i,j})$.

(2.5) Consider Median$(R_{i,j}^B)$ as the measure of centrality.

Algorithm 3: Detection Via Boostrapping the Geometric Mean

(3.3) Here, we take $R_{i,j} = \sqrt[n]{\prod_l (D_{l,i,j})}$.

(3.5) Consider Median$(R_{i,j}^B)$ as the measure of centrality.

Algorithm 4: Detection Via Closeness to Average

(4.2) Define $M_{i,j} = \inf_l |D_{l,i,j} - \frac{\sum_{l=1}^n D_{l,i,j}}{n}|$.

(4.5) Consider Median$(R_{i,j}^B)$ as the measure of centrality.

Algorithm 5: Pairwise Comparisons

(5.2) Calculate $u = 1, ..., \binom{n}{2}$ many distances $|D_{l,i,j} - D_{m,i,j}|$, where $1 \leq m < l \leq n$. Order the distances and label them as $M_{i,j}(1), ..., M_{i,j}(u)$ where $M_{i,j}(1)$ represents the smallest distance. Let $\Big(D_{l,i,j}(r), D_{m,i,j}(r)\Big)$ correspond to the pair of values belong to $M_{i,j}(r)$.

(5.3) If $n \geq 5$, let $R_{i,j} = \frac{\sum_{r=1}^{3}\Big(D_{l,i,j}(r)+D_{l,i,j}(r)\Big)}{6}$.

If $n = 4$, let $R_{i,j} = \frac{\sum_{r=1}^{2}\Big(D_{l,i,j}(r)+D_{l,i,j}(r)\Big)}{4}$.

If $n \leq 3$, let $R_{i,j} = \frac{\Big(D_{l,i,j}(1)+D_{m,i,j}(1)\Big)}{2}$.

(5.5) Consider Median$(R^B_{i,j})$ as the measure of centrality.

Note that the watermark sequence is inserted into a subset of the original signal. This fact can be used to improve the efficiency of the simulation. Specifically, the Bootstrap procedure could be performed on the entire set of K watermarked signal. However, this will increase the computational complexity of the simulation. Restricting our analysis to this subset will not change the results as the rest of the signal is not important to the watermark analysis. This restriction will greatly decrease the time required for each simulation.

3 Results

3.1 Simulation

To investigate the performance of the proposed algorithms, we perform a series of simulations. These simulations produce two outputs. The first output is the pointwise coverage of the confidence intervals. Our empirical findings are consistent with the Bootstrap theory [9,16]. Additionally, we determine the average span of each confidence interval. This output is used to determine which algorithms generate tighter bounds while still maintaining high coverage.

We organize the results around three parameters shared by all the algorithms: 1–the number of Bootstrap iterations B, 2–the sample size n, and 3–the number of watermarked copies K. Our simulations show that all algorithms remain consistent for different combinations of parameter values (B=50, 100, 1000, n=3, 5, 7). Additionally, we found that increasing K greatly decreases the average span and increases the coverage. Most importantly, the average span of the confidence intervals remains small while the coverage of the confidence intervals increases. This suggests that a collusion attack might be feasible with a small number of colluding documents.

We perform simulations to calculate the coverage and the average span of each confidence interval with $K = 10, 15, 20$ independently watermarked copies using sample sizes $n = 3, 5, 7$. Each simulation was repeated for $B = 50, 100, 1000$ iterations to ensure the stability of the result. The selected results are presented in the Appendix.

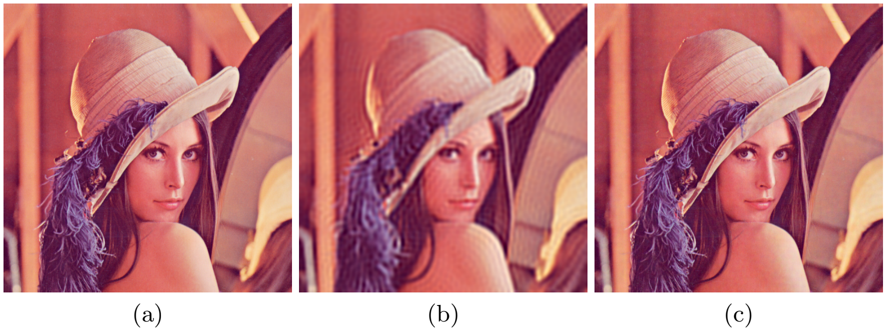

(a) (b) (c)

Fig. 1. (a) The original Lena picture (b) the significant parts (c) a watermarked version

We convert the "Lena" picture (Figure 1(a)) into a signal of DCT2 coefficients. We watermark the lowest frequency coefficients, excluding the constant coefficient. In Figure 1(b), we show the analog representation of the coefficients that have been watermarked. In Figure 1(c), we include the picture with the watermark inserted. In Figure 2, we demonstrate the reconstructed picture obtained via replacing each DCT2 coefficient with the median of algorithms 1,2, and 5 respectively.

To quantify the reconstructive quality of Figures 2(a), 2(b), and 2(c), we calculated the sequence W^* by reversing the watermarking process with the reconstructed signal D^* using

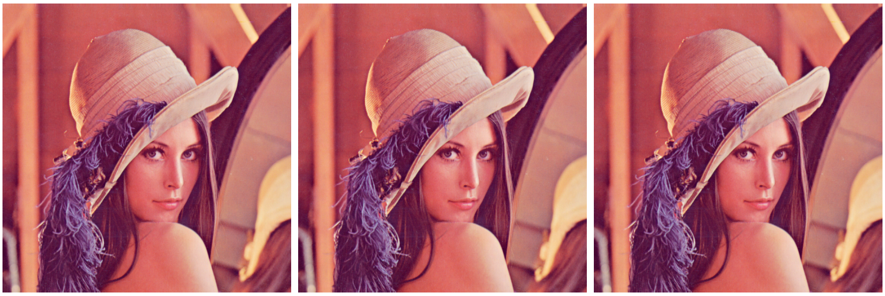
$$W^* = \frac{D^* - D_0}{\alpha}. \tag{6}$$

This recovered watermark is then compared to one of the original watermarks W_i, $1 \leq i \leq K$, using the similarity metric in equation (1). The similarity metric reveals that almost all of our algorithms perform the same. For example, comparing a watermarked image with itself using the above equations yield the result of 146.87. When this is done repeatedly, all results are clustered around $147.6 = \sqrt{21777}$ where 21777 is L, the length of the watermark. When two independently created watermarked pictures are compared using this algorithm, the results are clustered around zero [3]. When the median of the confidence intervals is used to construct W^*, then

$$\text{sim}(W_i, W^*) \approx \frac{147.6}{k},$$

where W_i is the watermark associated with one of the documents used to create W^*.

All of these simulations have been performed using the optimization described at the end of section 2 on freely available software. In terms of complexity, with a watermark of size L on a Bootstrap of B repetitions per signal and n samples per repetition, the algorithm runs in time $O(LBn)$.



Algorithm 1 Algorithm 2 Algorithm 5

Fig. 2. The reconstructed image produced from 100 Bootstrap iterations with a sample size of 7 and a population of 20 using algorithms 1, 2, and 5

3.2 Numerical Results

The tables in the Appendix detail the output of the previously described algorithms. Each table represents a different algorithm with an indicated number of watermarked pictures as an input. Tables have not been included for algorithms three and four because these algorithms produced results that were indistinguishable from algorithm one. Each algorithm was tested with 10, 15, and 20 watermarked documents. Within each table, the columns are organized in categories of 50, 100, and 1000 Bootstrap iterations and subcategories of 3, 5, and 7 samples per Bootstrap iteration. Each row indicates the percentile confidence interval with confidence levels 30, 50, 70, 80, 90, 95, and 99. For example, in table A1, these results refer to algorithm one performed with ten independently watermarked documents. In column 100, sub-column 5, and row 90, we see that the 90 percentile confidence interval contained the correct value 92.9% of the time and the average span of the intervals was 1.26. For a formal definition of the percentile confidence interval, see equation (3).

All of the algorithms perform well even with a small number of watermarked pictures and a low number of Bootstrap iterations. However, there are differences worth discussing. The primary result concerning the coverage is that the number of Bootstrap iterations quickly leads to diminishing returns. Running the Bootstrap procedure 1000 times does not yield better results than doing it 100 times. Most differences are much less than 10%. The difference between 100 Bootstrap iterations and 50 iterations is greater, but low iteration results are still impressive.

The number of independently watermarked documents does play a large role in the process. The most significant differences occur at the lower confidence intervals. For example, with 10 watermarked documents, using algorithm 1 with 100 Bootstrap iterations and a sample size of 5, the middle half of the samples contained the true value 59.6% of the time. In the same situation with 20 documents, the result improves to 75.8%.

The span also varies greatly with the different parameters of each algorithm. The results were similar to those of the percentage coverage, so there will not be as great an emphasis on each of the different effects of the parameters. The most important result is that the number of Bootstrap iterations does not change the span much and that the size of the population of watermarked pictures does have a positive correlation with the span.

Clearly, the precision of the result will decrease as the percentile confidence interval increases as there is a positive correlation with average span. Pragmatically, one would like to strike a balance between the percentile confidence interval and the precision. As shown in table A9, at lower percentile confidence intervals with small span, algorithm 5 still produces satisfactory results. This trend continues to hold with fewer Bootstrap iterations and smaller numbers of watermarked documents.

In the end, the objective is to get a high coverage and a low span. To quantify this goal we look at the ratio of the coverage to the span. Referencing the tables in the Appendix, it is clear upon inspection that algorithm 1 produces the greatest ratio of coverage to span. This ratio increases greatly with the population of watermarked pictures because algorithm 1 produces smaller spans with larger populations of watermarked pictures in each Bootstrap iteration, and larger samples will produce a better ratio. For specific examples refer to table A7.

The median of the outputs from each Bootstrap iteration forms the reconstructed image. When watermarked, the PSNR of an image is about 38.4dB. After the process is completed all algorithms produce an image with PSNR above 40dB. These PSNR values are close to the values obtained from the simple mean or median algorithm. However, using the similarity metric, we find that our proposed methods perform at least as good as the simple mean or median

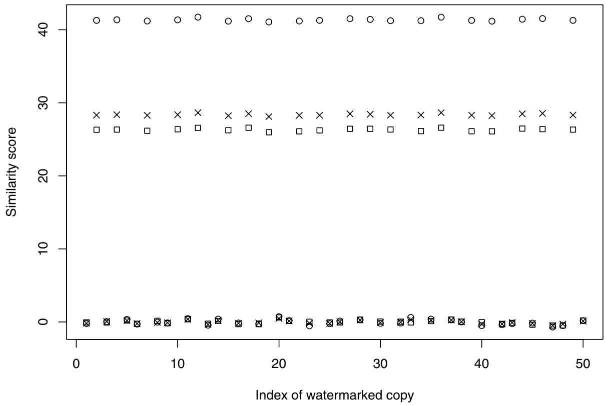


Fig. 3. Similarity values of the reconstructed image with 20 colluding images when the noise distribution is skewed to the right. The similarity values for the image reconstructed with the simple mean are represented with circles. The similarity values for the image reconstructed with the median are represented with crosses. The similarity values for the image reconstructed with algorithm 5 are represented with boxes. The similarity values with 30 other independently watermarked images are shown for comparison.

attacks. For example, when the colluders possess images in which some of the watermarked signals contain outliers, our method improves the similarity results significantly. This is due to the fact that our proposed statistical procedure is non-parametric. Consequently, the Bootstrap-based algorithms perform better than the median or the mean attack specifically in the presence of extreme values [9]. To elaborate, we generate noise from a right-skewed Gamma$(1, 1/5)$ distribution. Under this scenario, the attack in algorithm 5 improves the similarity metric by approximately 36% when compared to the mean and by approximately 7% when compared to the median (see figure 3).

4 Discussion

In this paper, we used the simple equal-tail two-sided Bootstrap percentile confidence interval which has second order accuracy. There have been theoretical improvements to this method. Examples include the Bootstrap bias-corrected percentile, the Bootstrap accelerated bias-corrected percentile and the hybrid Bootstrap confidence intervals (See Shao [16] for a comprehensive theoretical discussion).

We demonstrate the use of resampling to acquire measures of variation associated with a collusion. One should recognize that in principle, the reconstruction of the original document can be done with simple averaging [3]. However, in the process of Bootstrap, we obtain more information about the original unwatermarked document. We believe that Bootstrap resampling methods presented here could open up avenues of research by providing methods for quantifying the probability that the signal is contained in a given interval. This will allow researchers to use many nonparametric methods, resampling being only one of them.

We observe that the Bootstrap works with small sample sizes, low number of iterations and modest number of watermarked copies. Most importantly, while keeping a high PSNR value, the Bootstrap method lowers the value of the similarity metric in comparison with frequently used attacks such as the mean and the median. These are reassuring results. We discuss the statistical inference using the multiple testing adjustment and the use of alternative confidence intervals in another paper. We are currently investigating the use of the Bootstrap on collusion attacks on watermarked motion pictures, where there is a high amount of signal redundancy. We conjecture that the Bootstrap methodology will be more effective at reconstructing the original signal in these cases.

Acknowledgements

We would like to thank Christopher Gutierrez, Joseph Sutton, and Max Velado for their help in the preparation of this paper.

References

1. Cannons, J., Moulin, P.: Design and Statistical Analysis of a Hash-Aided Image Watermarking System. IEEE Transactions on Image Processing 13(10), 1393–1408 (2004)
2. Comesaña, P., Pérez-Freire, L., Pérez-González, F.: The Return of the Sensitivity Attack. In: Barni, M., Cox, I., Kalker, T., Kim, H.-J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 260–274. Springer, Heidelberg (2005)
3. Cox, I.J., Kilian, J., Leighton, F.T., Shamoon, T.: Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing 6(12), 1673–1687 (1997)
4. Delaigle, J.F., De Vleeschouwer, C., Macq, B.: Watermarking algorithm based on a human visual model. Signal Processing 66, 319–335 (1998)
5. Dong, P., Brankov, J.G., Galatsanos, N.P., Yang, Y., Davoine, F.: Digital Watermarks Robust to Geometric Distortions. IEEE Transactions on Image Processing 14(12), 2140–2150 (2005)
6. Doërr, G., Dugelay, J.: Countermeasures for Collusion Attacks Exploiting Host Signal Redundancy. In: Barni, M., Cox, I., Kalker, T., Kim, H.-J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 216–230. Springer, Heidelberg (2005)
7. Doërr, G., Dugelay, J.: Security Pitfalls of Frame-by-Frame Approaches to Video Watermarking. IEEE Transactions on Signals Processing 52(10), 2955–2964 (2004)
8. Efron, B.: The Jackknife, the Bootstrap and other Resampling Plans. Society for Industrial and Applied Mathematics, Philadelphia (1982)
9. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall, New York (1993)
10. Ergun, F., Kilian, J., Kumar, R.: A Note on the Limits of Collusion-Resistant Watermarks. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 140–149. Springer, Heidelberg (1999)
11. Furon, T.: A Survey of Watermarking Security. In: Barni, M., Cox, I., Kalker, T., Kim, H.-J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 201–215. Springer, Heidelberg (2005)
12. Khayam, S.A.: The Discrete Cosine Transform (DCT): Theory and Application. Michigan State University (2003)
13. Kiyavash, N., Moulin, P.: A Framework for Optimizing Nonlinear Collusion Attacks on Fingerprinting Systems. In: 40th Annual Conference on Information Sciences and Systems, pp. 1170–1175 (2006)
14. Kiyavash, N., Moulin, P.: On Optimal Collusion Strategies for Fingerprinting. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 5, p. V (2006)
15. Podilchuk, C.I., Delp, E.J.: Digital Watermarking: Algorithms and Applications. Signal Processing Magazine 18(4), 33–46 (2001)
16. Shao, J., Tu, D.: The Jackknife and Bootstrap. Springer, New York (2005)
17. Soriano, M., Fernandez, M., Cotrina, J.: Fingerprinting Schemes: Identifying the Guilty Sources Using Side Information. In: Barni, M., Cox, I., Kalker, T., Kim, H.-J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 231–243. Springer, Heidelberg (2005)
18. Vila-Forcén, J.E., Voloshynovskiy, S., Koval, O., Pérez-González, F., Pun, T.: Practical Data-Hiding: Additive Attacks Performance Analysis. In: Barni, M., Cox, I., Kalker, T., Kim, H.-J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 244–259. Springer, Heidelberg (2005)

19. Vinod, P., Bora, P.K.: A New Inter-Frame Collusion Attack and a Countermeasure. In: Barni, M., Cox, I., Kalker, T., Kim, H.-J. (eds.) IWDW 2005. LNCS, vol. 3710, pp. 147–157. Springer, Heidelberg (2005)
20. Wang, Z., Wu, M., Zhao, H., Liu, K.: Resistance of Orthogonal Gaussian Fingerprints to Collusion Attacks. In: Proceedings of International Conference on Multimedia and Expo., vol. 1, pp. 617–620 (2003)
21. Wolfgang, R.B., Podilchuk, C.I., Delp, E.J.: Perceptual Watermarks for Digital Images and Video. Proceedings of IEEE 87(7), 1108–1126 (1999)
22. Wu, M., Trappe, W., Wang, Z., Liu, K.: Collusion-resistant Fingerprinting for Multimedia. IEEE Signal Processing Magazine 21(2), 15–27 (2004)
23. Zhao, H., Wu, M., Wang, Z., Liu, K.: Forensic Analysis of Nonlinear Collusion Attacks for Multimedia Fingerprinting. IEEE Transactions on Image Processing 14(5), 646–661 (2005)
24. Zhao, H., Wu, M., Wang, Z., Liu, K.: Nonlinear Collusion Attacks on Independent Fingerprints for Multimedia. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 5, pp. 664–667 (2003)
25. Zhu, W., Xiong, Z., Zhang, Y.: Multiresolution Watermarking for Images and Video. IEEE Transactions on Circuits and Systems for Video Technology 9(4), 545–550 (1999)

Appendix: Selected Results

Table A1. The table of coverage and span for the Mean Algorithm (1) with a population of size 10. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			100			1000		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.467,.386	.376,.282	.312,.226	.472,.378	.378,.280	.321,.222	.481,.376	.373,.272	.319,.217
50	.703,.682	.594,.506	.514,.415	.719,.691	.596,.513	.514,.418	.722,.690	.601,.515	.522,.419
70	.869,1.07	.775,.804	.699,.664	.882,1.07	.788,.800	.709,.660	.891,1.07	.794,.803	.717,.663
80	.932,1.32	.851,.995	.790,.828	.936,1.32	.862,.995	.793,.828	.942,1.33	.866,1.00	.801,.827
90	.969,1.62	.921,1.24	.867,1.04	.972,1.65	.929,1.26	.879,1.05	.977,1.68	.936,1.28	.891,1.06
95	.983,1.98	.956,1.53	.923,1.29	.983,1.93	.959,1.49	.926,1.25	.988,1.96	.966,1.51	.933,1.27
99	.989,2.12	.968,1.65	.940,1.38	.991,2.19	.975,1.71	.952,1.44	.996,2.46	.986,1.92	.971,1.62

Table A2. The table of coverage and span for the Median Algorithm (2) with a population of 10. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			100			1000		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.458,.465	.376,.356	.318,.287	.461,.459	.350,.321	.281,.243	.466,.465	.244,.201	.255,.200
50	.698,.814	.610,.648	.544,.562	.698,.795	.634,.667	.571,.597	.657,.696	.657,.693	.645,.695
70	.872,1.28	.788,1.02	.713,.851	.881,1.27	.805,1.04	.717,.831	.886,1.25	.855,1.17	.649,.707
80	.932,1.60	.873,1.27	.814,1.08	.933,1.61	.883,1.26	.834,1.13	.943,1.69	.889,1.25	.886,1.26
90	.969,2.02	.936,1.61	.893,1.37	.978,2.01	.942,1.64	.900,1.36	.979,1.94	.966,1.83	.888,1.26
95	.987,2.47	.969,2.02	.945,1.72	.990,2.45	.971,1.94	.945,1.67	.992,2.78	.978,1.94	.968,1.89
99	.992,2.64	.978,2.17	.962,1.86	.996,2.81	.984,2.26	.971,1.94	.999,3.01	.996,2.82	.977,1.99

Table A3. The table of coverage and span of the Pairwise Comparison Algorithm (5) with a population of size 10. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			100			1000		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.596,.695	.502,.442	.511,.466	.605,.687	.512,.435	.525,.458	.650,.689	.524,.431	.528,.450
50	.833,1.22	.747,.788	.756,.822	.850,1.21	.761,.786	.771,.826	.884,1.24	.770,.793	.780,.825
70	.952,1.89	.902,1.25	.910,1.30	.967,1.88	.921,1.24	.925,1.30	.977,1.93	.928,1.25	.930,1.30
80	.983,2.37	.952,1.55	.961,1.62	.986,2.38	.964,1.55	.963,1.62	.983,2.27	.967,1.56	.972,1.62
90	.995,2.80	.982,1.98	.984,2.06	.998,2.93	.986,2.00	.987,2.08	.999,3.01	.989,2.04	.990,2.11
95	.998,2.95	.993,2.43	.993,2.49	.998,3.00	.994,2.41	.995,2.47	.999,3.01	.997,2.44	.998,2.51
99	.998,2.99	.996,2.60	.996,2.64	.999,3.01	.996,2.75	.997,2.78	.999,3.01	.999,3.00	.999,3.00

Table A4. The table of coverage and span for the Mean Algorithm (1) with a population of 15. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			100			1000		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.549,.428	.456,.330	.396,.277	.567,.429	.468,.330	.403,.278	.582,.430	.472,0.331	.407,0.276
50	.807,.748	.701,.576	.625,.484	.821,.749	.713,.577	.636,.485	.838,.752	.723,0.578	.641,.487
70	.943,1.14	.878,.880	.819,.741	.951,1.15	.887,.882	.827,.743	.958,1.15	.897,0.885	.837,.746
80	.983,1.41	.954,1.08	.923,.915	.987,1.41	.960,1.09	.930,.916	.989,1.42	.966,1.09	.937,.920
90	.992,1.77	.973,1.38	.951,1.16	.994,1.79	.978,1.39	.957,1.17	.996,1.80	.982,1.39	.962,1.18
95	.997,2.19	.989,1.71	.977,1.45	.998,2.10	.990,1.63	.978,1.38	.998,2.11	.992,1.64	.983,1.39
99	.997,2.18	.989,1.71	.977,1.45	.999,2.29	.994,1.79	.985,1.52	.999,2.44	.996,1.92	.992,1.62

Table A5. The table of coverage and span for the Median Algorithm (2) with a population of 15. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			100			1000		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.544,.472	.456,.370	.400,.306	.554,.464	.455,.356	.405,.299	.555,.469	.401,.287	.392,.280
50	.785,.826	.693,.657	.630,.553	.799,.835	.698,.653	.635,.566	.827,.884	.690,.618	.688,.616
70	.920,1.30	.866,1.03	.813,.877	.946,1.31	.890,1.03	.835,.888	.962,1.37	.882,.980	.877,.967
80	.969,1.62	.935,1.28	.895,1.10	.977,1.63	.941,1.29	.904,1.10	.982,1.68	.961,1.36	.890,1.01
90	.990,2.06	.974,1.64	.953,1.40	.994,2.08	.977,1.65	.960,1.40	.994,2.09	.987,1.77	.963,1.37
95	.997,2.54	.990,2.05	.979,1.75	.998,2.48	.991,1.97	.980,1.69	.9992,2.45	.993,1.90	.989,1.81
99	.998,2.73	.993,2.21	.987,1.90	.9996,2.88	.997,2.31	.991,1.97	1.00,3.40	.999,2.50	.998,2.31

Table A6. The table of coverage and span of the Pairwise Comparison Algorithm (5) with a population of 15. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			100			1000		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.682,.712	.583,.437	.589,.455	.695,.697	.593,.430	.603,.450	.704,.661	.607,.426	.608,.439
50	.901,1.23	.821,.779	.826,.807	.915,1.24	.840,.781	.846,.812	.934,1.27	.851,.782	.854,.810
70	.982,1.93	.951,1.23	.951,1.28	.988,1.93	.963,1.24	.964,1.27	.991,1.86	.971,1.23	.972,1.28
80	.995,2.39	.982,1.54	.981,1.59	.997,2.39	.987,1.54	.988,1.59	.999,2.47	.989,1.54	.989,1.59
90	.999,2.93	.994,1.96	.994,2.03	.9997,3.07	.996,1.98	.998,2.05	1.00,3.37	.997,2.01	.998,2.07
95	.9997,3.25	.998,2.47	.998,2.52	1.00,3.34	.999,2.39	.999,2.46	1.00,3.41	.9992,2.43	.9996,2.51
99	1.00,3.34	.999,2.66	.999,2.71	1.00,3.40	.9997,2.79	.9995,2.84	1.00,3.41	1.00,3.30	1.00,3.31

Table A7. The table of coverage and span for the Mean Algorithm (1) with a population of size 20. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			**100**			**1000**		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.606,.396	.500,.292	.430,.235	.622,.392	.508,.287	.434,.232	.630,.385	.510,.283	.434,.229
50	.853,.702	.754,.527	.667,.433	.859,.713	.758,.533	.671,.436	.873,.712	.770,.535	.685,.438
70	.961,1.11	.908,.840	.849,.695	.973,1.10	.918,.835	.866,.690	.978,1.11	.928,.835	.877,.692
80	.985,1.37	.959,1.04	.917,.866	.991,1.37	.963,1.04	.925,.865	.993,1.38	.968,1.04	.931,.865
90	.996,1.71	.983,1.31	.961,1.09	.998,1.73	.988,1.32	.969,1.10	.998,1.76	.992,1.34	.977,1.12
95	.999,2.10	.994,1.62	.982,1.35	.9991,2.04	.996,1.57	.988,1.31	.9994,2.07	.998,1.59	.992,1.33
99	.9996,2.25	.997,1.74	.990,1.46	.9997,2.34	.998,1.82	.994,1.52	1.00,2.63	.9995,2.04	.999,1.72

Table A8. The table of coverage and span for the Median Algorithm (2) with a population of 20. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			**100**			**1000**		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.602,.475	.508,.367	.446,.307	.612,.463	.515,.360	.455,.305	.613,.463	.491,.327	.486,.320
50	.834,.828	.754,.654	.695,.558	.848,.831	.759,.658	.693,.554	.876,.841	.741,.611	.719,.574
70	.957,1.31	.912,1.03	.870,.881	.969,1.31	.927,1.03	.884,.881	.978,1.39	.942,1.08	.881,.846
80	.988,1.63	.963,1.29	.934,1.10	.991,1.62	.968,1.29	.937,1.10	.994,1.66	.974,1.32	.951,1.12
90	.996,2.06	.987,1.64	.972,1.40	.998,2.08	.992,1.65	.979,1.41	.9993,2.17	.996,1.73	.987,1.43
95	.9992,2.57	.996,2.05	.990,1.76	.9996,2.48	.997,1.99	.991,1.69	.9998,2.60	.999,2.04	.997,1.76
99	1.00,2.77	.998,2.22	.997,1.90	1.00,2.88	.9992,2.31	.996,1.97	1.00,3.39	1.00,2.67	.9993,2.23

Table A9. The table of coverage and span of the Pairwise Comparison Algorithm (5) with a population of size 20. Span is indicated second and is expressed in terms of multiples of the standard deviation.

	50			**100**			**1000**		
	3	**5**	**7**	**3**	**5**	**7**	**3**	**5**	**7**
30	.722,.714	.625,.431	.639,.448	.752,.700	.645,.428	.656,.441	.777,.687	.661,.420	.659,.430
50	.935,1.24	.867,.768	.869,.793	.949,1.24	.883,.771	.882,.793	.957,1.24	.899,.771	.897,.794
70	.990,1.93	.969,1.22	.971,1.25	.996,1.94	.978,1.22	.977,1.26	.997,1.92	.985,1.22	.985,1.26
80	.998,2.39	.991,1.51	.992,1.57	.9993,2.40	.995,1.52	.994,1.57	.9995,2.38	.996,1.52	.996,1.57
90	.9998,2.98	.997,1.93	.998,2.00	1.00,3.07	.999,1.95	.999,2.01	1.00,3.05	.9994,1.97	.9997,2.04
95	1.00,3.39	.9993,2.44	.9992,2.50	1.00,3.48	.9994,2.35	.9998,2.42	1.00,3.66	.9997,2.38	.9998,2.46
99	1.00,3.52	.9994,2.64	.9997,2.70	1.00,3.63	.9998,2.77	1.00,2.82	1.00,3.66	1.00,3.22	1.00,3.27

Improved Data Hiding Technique for Shares in Extended Visual Secret Sharing Schemes

Rabia Sirhindi, Saeed Murtaza, and Mehreen Afzal

College of Signals, National University of Sciences and Technology, Pakistan
msis-5.rabia@mcs.edu.pk, smurtaza-mcs@nust.edu.pk,
mehreenafzal@hotmail.com

Abstract. An improved data hiding technique is proposed in this paper to hide the shares of a secret image in an Extended Visual Secret Sharing (EVSS) scheme. It is based on Least Significant Bit (LSB) substitution with a little modification that the embedding capacity varies with each pixel of the host image and depends upon the surrounding pixels' color difference. This not only increases the embedding capacity of the host image as compared to simple LSB substitution but also yields high Peak Signal to Noise Ratio (PSNR) values for host and stego images. Moreover, results indicate that the proposed data hiding process improves the security of camouflage images in EVSS schemes since shares are completely hidden in the cover images without any trace of their presence, unlike most of the previous share hiding approaches, thus preventing shares from alteration during transmission.

Keywords: Visual cryptography, extended visual secret sharing, data hiding, LSB encoding, adaptive LSB encoding.

1 Introduction

Steganography is the art of hiding secret messages in apparently harmless carrier messages such that the very existence of the secret is concealed. Unlike cryptography, which uses codes and ciphers to change the structure of secret information, steganography uses data hiding techniques rendering the message invisible. Use of steganographic methods on cryptographically treated secret messages provides an added layer of security since an enciphered message might arouse suspicion where as a camouflaged and invisible message does not.

Visual cryptography, first introduced by Naor et al. in [1], is a new method of encrypting data which is taken in the form of black and white images i.e., pictures, text, handwriting, etc. The idea is based upon secret sharing schemes with the exception that now the secret data is an image. A dealer divides the secret image among n number of participants where it can only be recovered when k or more participants ($k \leq n$) stack their respective pieces together. Any less than k participants gather no information about the secret when there pieces are stacked together. These pieces of the secret data (here an image) are called shares and the phenomena is called Visual Secret Sharing (VSS) as shown in

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 376–386, 2008.

Fig. [1]. The image becomes visible when k participants stack the transparencies containing shares and the decryption is performed by the human visual system. The basic concept of visual cryptography proposed in [1] is applicable to black and white images. A number of extensions have been proposed, though, which also deal with grayscale and color images [3,4,5,8]. In addition, schemes have been developed that share and hide a secret image in multiple significant cover images [6,7,8]. The shares are, thus, camouflaged in meaningful but innocent-looking cover images possessing no trace of the original secret information. This form of VSS is called Extended Visual Secret Sharing (EVSS)[2]. It improves the security of secret image because now the random looking black and white content of the share images is concealed in colourful images.

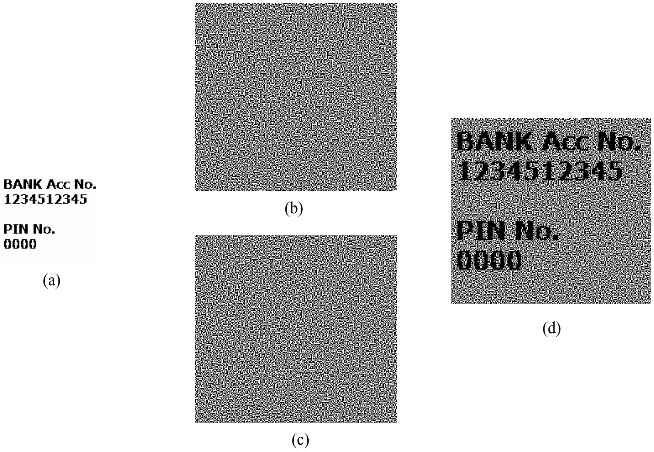


Fig. 1. (2, 2) visual secret sharing. (a) Original Image I of size 120x120, (b)-(c) Shares SH1 and SH2 of size 240x240, (d) Recovered image RecI by stacking SH1 and SH2.

This article proposes a data hiding technique that can be employed in an EVSS scheme to hide secret image shares in meaningful color images. It uses adaptive LSB substitution in which the number of bits to be hidden is calculated individually for each pixel and is based on the difference in surrounding pixel values. The purpose is to ensure that variances introduced by embedded data in the solid colour areas of host image do not become noticeable. Therefore, areas with little difference in pixel intensities are packed with lesser number of share bits than those with greater difference values. The prime advantage of using a different data hiding method is that it gives improved PSNR values over the previous techniques. The stego images produced using this technique, do not disclose any information about the hidden data that becomes noticeable to human eyes. This is especially useful in visual authentication applications [9], [10] (such as document authentication, etc.) since it prevents the shares from man-in-the-middle attacks. Not only is the information now encoded in cover image, but also it is scrambled enough to resist an interception-modification attack as described in [10].

The remaining paper is structured as follows. Section 2 provides an insight into some existing EVSS schemes and share hiding technique used in these schemes. Section 3 covers a detailed description of data hiding in images by simple LSB and adaptive LSB substitution. This followed by the proposed data hiding approach in section 4, which is based on adaptive LSB and calculates the number of bits individually for each pixel in the host image. Section 5 presents some analysis and results based on Peak Signal to Noise Ratio (PSNR) values of stego and host images and finally section 6 concludes the article.

2 Extended Visual Secret Sharing Schemes and Share Hiding Principle

Some advanced secret sharing schemes will be seen in this section based on *modified visual cryptography*, which use significant images called cover or shadow images to hide the secret image. These innocent looking images reveal no information about the original image, until they are stacked. The stacking operation constructs an image that is neither of the two covers. This form of visual cryptography also called extended visual cryptography is more secure since the confusing and meaningless copies of shares produced in basic VSS schemes may invite attempted attack. The following subsections briefly discuss some of the existing EVSS schemes.

2.1 Chang and Chen's Scheme

This secret sharing scheme generates two shares of the colored secret image I such that each pixel is expanded into two txt sub-pixel blocks (where M=txt is bounded by the number of colors in the secret image), one for each share, which is subsequently hidden in the cover image pixels. It uses a Color Index Table (CIT) that stores each color present in the secret image against an index or code value which is used to successfully retrieve this color information when reconstructing the original image from two camouflage images. A detailed description on how the shares are created can be found in [6].

2.2 Chang and Yu's Scheme

Chang et al. provide an algorithm to share secret gray images in multiple images in the scheme proposed in [7]. Each pixel's gray value in secret image is encoded into an 8-bit binary string k = $(k_1k_2k_3 \cdots k_8)$. Two pixel shares are created from this binary value, producing two *3*x*3* sub-pixel blocks for the original pixel, one for each participant. After repeating the process for all pixel in the secret image, the two shares constructed are of size *3M*x*3N* (where M x N is the size of original image). The secret image is recovered using XOR operation between corresponding sub-pixel blocks in binary shares hidden in the two camouflage images. More detail on sharing and recovering algorithms can be seen in [7].

2.3 Hiding Technique Used in Previous Schemes

Our main focus in this paper is on the share hiding techniques used to hide data contained in binary shares produces as a result of applying secret sharing schemes. In the previously described two schemes, once the sub-pixel blocks of original image pixel are obtained, these are hidden in the cover images by simply filling colors k_1 and k_2 from corresponding pixels of covers O_1 and O_2. This is done in such a way that only the sub-pixel locations containing a 1 (color black in share) are filled with the color of corresponding pixel in cover image. The phenomenon is illustrated in Fig. 2. Thus, the process of hiding a share in cover image is fairly simple and straight forward. It does not involve any encoding of data in the host image, rather fills the sub-pixel blocks in the binary share with colors from corresponding cover image pixels in the spatial domain.

3 Data Hiding Techniques

Data hiding techniques are employed to conceal the existence of secret data. The secret information can be embedded in digital media (text, images, video, audio, etc) in such a way that its presence is barely discernible to human perception. Data hiding techniques have gained significant interest in recent years and a number of techniques have been proposed for various applications such as tamper proofing, copyright protection, authentication and watermarking. In this article it is used in the context of VSS schemes and authentication applications involving visual cryptography. The following subsection covers some very basic data hiding techniques.

3.1 Least Significant Bit (LSB) Encoding

The simplest of all data hiding techniques is the Least Significant Bit (LSB) encoding or substitution. It uses the LSB plane of the cover or container image to hide secret data. The secret message is embedded into the k rightmost least significant bits of the original image. If the cover image is grayscale, each pixel value can be represented as an 8-bit binary value. The secret message (text, image, etc) is first converted to a binary bit stream and then k bits are read at one time to be embedded into given cover image pixel $p_{i,j}$ as follows,

$$p'_{i,j} = p_{i,j} - p_{i,j} \; mod \; 2^k \; \oplus \; m \tag{1}$$

This process is repeated until all bits have been embedded into cover image where the number of secret bits per pixel is fixed for each pixel. The quality of the stego image produced as a result of LSB substitution depends upon number of bits embedded per pixel, where the former degrades as the latter increases. Results on PSNR values for 1 to 5 bits per pixel are given in [11].

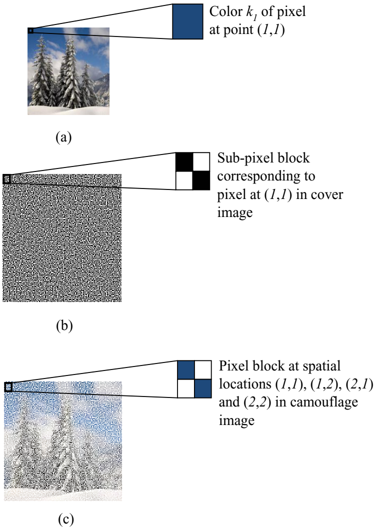


Fig. 2. Hiding share in cover image by simple color filling technique. (a) Cover image C of size 100x100, (b) Share SH1 of size 200x200, (c) Camouflage image C' with share SH1 hidden.

4 Proposed Data Hiding Technique to Hide Shares

In this section, an Adaptive LSB (A-LSB) substitution technique is proposed for hiding secret image shares in cover images. It differs from simple LSB in the way number of bits to be replaced in each cover image pixel is calculated. Data hiding using neighborhood pixel information is proposed in [12] where it works along with Optimal Pixel Adjustment Process (OPAP) because the original scheme is unable to extract the hidden data completely from stego image. Here, however, OPAP is not performed which improves the computation time and the data is recovered successfully in its original form also. The reason LSB is chosen for hiding is that data has to be recovered properly and losslessly since the shares are required to reconstruct the secret image.

An RGB container image C of size MxN is taken and its green plane C_g is extracted for the data embedding. This is because human eyes are less sensitive to green as compared to red and blue. 65% receptor cones over the retinal surface of human eyes are sensitive to red, 33% to green and only 2% to blue; however these are the most sensitive [13]. If data is embedded in all three color channels of the image, the noise becomes visible to human eyes and quality of stego image deteriorates. Each share is a monochrome image with pixel values equal to 0 or 1. This two dimensional image is converted to one dimensional array of bits that are subsequently embedded in the cover image. The following subsections discuss hiding and extraction process in detail.

4.1 Embedding Algorithm

The embedding process is similar to that of LSB substitution except for one additional step. Once C_g is extracted from the cover image, the number of bits for each green pixel is calculated as a logarithm of difference between its upper and left neighbors. This difference value denoted by G_d, represents the change in color intensities from one adjacent pixel to another. Based on this value, the number of bits to be embedded is calculated for each pixel(Equation 2). This method ensures that more data is embedded in the noisy regions of the host image than those have monotone color patterns.

$$G_d = pUpper - pLeft \tag{2}$$

The number of potential storage bits N_b is decided based on G_d. It can range from 1 to 4. If bit positions beyond four LSBs are used, the quality of stego image degrades. Moreover, four bits are embedded only in noisy areas of the cover image where the effect is unnoticeable.

$$N_b = 1 \quad \text{if } -1 \leq G_d \leq 1 \tag{3}$$

$$N_b = log_2(G_d) \quad \text{if } G_d \geq |1| \tag{4}$$

After calculating number of bits for the pixel, N_b bits from the data to be hidden are embedded in the pixel using Equation 1. When all data bits from the share are embedded in the cover image, the original green component C_g of cover image is replaced with the new green channel C'_g. This forms the stego image *C'*.

4.2 Extraction Algorithm

To extract data from stego image, the same procedure is repeated again. First the green component containing hidden information is extracted from the stego image *C'*. Potential number of secret data bits is calculated for each pixel in C'_g using Equations 3 and 4. Then, N_b rightmost bits of the pixel are extracted from the 8-bit binary string. When data from all pixels is calculated, it is converted back from bit stream to a two dimensional form of a share.

5 Experimental Results

Some results of embedding secret image shares in color images are presented in this section. A black and white secret image of dimensions 100x100 is shared using a (2,2) VSS scheme to generate two shares SH1 and SH2 of size 200x200 pixels each. These are hidden in the two cover images C1 (Pepper) and C2 (Hills) as shown in Fig. 3(a) and (b), using two data hiding techniques. The first set of images (Fig. 3(g) and (h)) are obtained using the proposed data hiding approach based on pixel value differencing. The second set (Fig. 3(c) and (d)) is produced as a result of applying the previous data hiding technique as given in

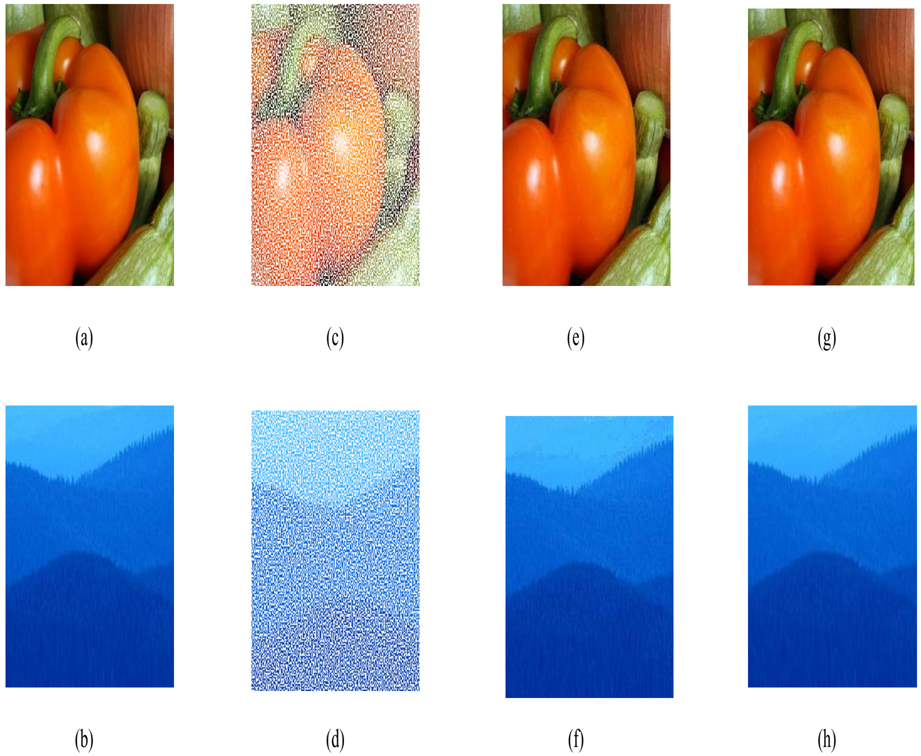


Fig. 3. Hiding SH1 and SH2 in cover images (a)-(b) Original cover images C1 and C2 of size 200x200, (c)-(d) Camouflage images C1' and C2' obtained using previous coloring technique, (e)-(f) Stego images of size 200x200 obtained from LSB substitution with 4 bits per pixel,(g)-(h) Stego images of size 200x200 obtained from A-LSB substitution

Table 1. Hiding Capacity and PSNR values for proposed data hiding scheme with share size of 200x200 pixels

Image	Image Size	Embedded	Hiding	PSNR
		Data Size	Capacity	(in dB)
		in bits)	in bits)	
Lena.jpg	256x256	40000	131472	45.4707
Baboon.jpg	256x256	40000	197909	41.0476
Hills.jpg	256x256	40000	78473	48.8608
Autumn.jpg	256x256	40000	189739	41.7013
Pepper.jpg	256x256	40000	103197	45.5479
Bridge.jpg	256x256	40000	191964	41.2171

section 2.3 of this article. A scaled down version of the cover images are required for the second method as they are later expanded to form camouflage images of size 200x200.

The noise in the second set of camouflage images can easily be seen without any zooming in, clearly indicating the presence of some hidden information in the image. Although the image is still significant, the quality is much deteriorated. On the contrary, stego images generated from the proposed data hiding solution appear very similar to the original covers. This is further supported by the PSNR values for six different images in Table 1. Hiding capacity refers to the number of bits that can be hidden in the image using A-LSB and PSNR is the Peak Signal to Noise Ratio of the stego-images obatined.

6 Discussion

The results in Table 1 show that employing A-LSB for data hiding of shares produced in a VSS scheme achieves high PSNR values and larger embedding capacity as compared to existing schemes. The table below shows the data hiding capacity for different images of size 256x256 pixels. This definitely is better than simple LSB where a fixed number of bits are embedded in all pixels irrespective of their location in the image i.e., whether they are in a solid fill area or a noisy one. PSNR values greater than 40 dB prove that the quality of original image is preserved.

This scheme is better than the others from a security point of view also because stego images produced reveal no information about hidden data as compared to previous technique (Fig. 3(c) and (d)). Even when compared to simple LSB

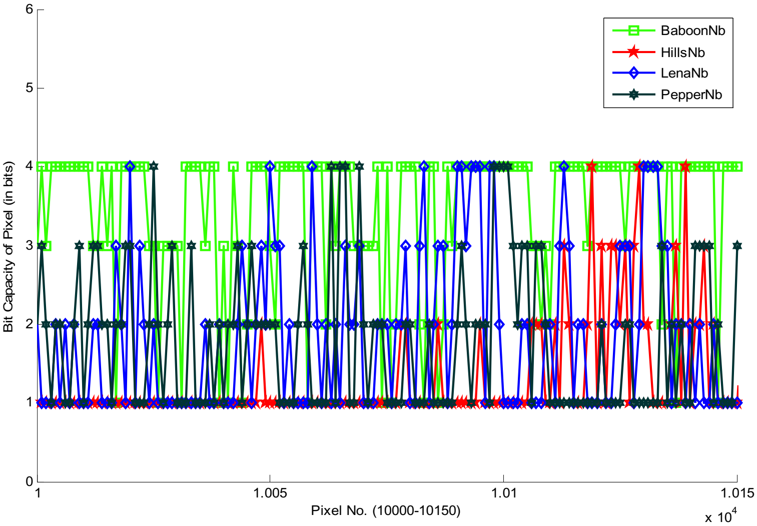
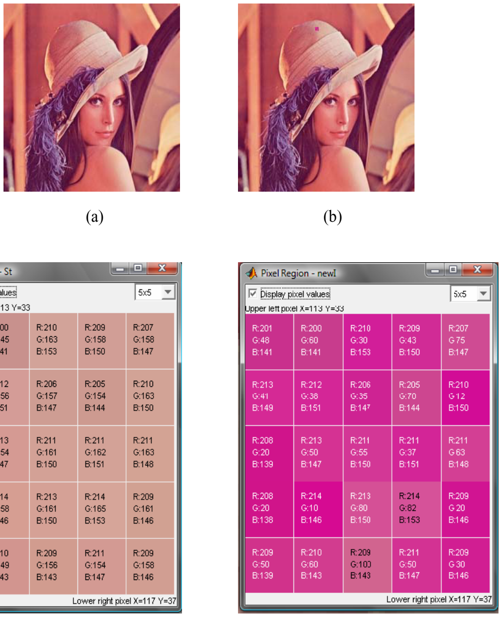


Fig. 4. Distribution of bits in four different host images for 150 randomly chosen pixels



(a)

(b)

Pixel Region - St

☑ Display pixel values 5x5

Upper left pixel X=113 Y=33

R:201 G:151 B:141	R:200 G:145 B:141	R:210 G:163 B:153	R:209 G:158 B:150	R:207 G:158 B:147
R:213 G:158 B:149	R:212 G:156 B:151	R:206 G:157 B:147	R:205 G:154 B:144	R:210 G:163 B:150
R:208 G:145 B:139	R:213 G:154 B:147	R:211 G:161 B:150	R:211 G:162 B:151	R:211 G:163 B:148
R:208 G:149 B:138	R:214 G:158 B:146	R:213 G:161 B:150	R:214 G:165 B:153	R:209 G:161 B:146
R:209 G:147 B:139	R:210 G:149 B:143	R:209 G:156 B:143	R:211 G:154 B:147	R:209 G:158 B:146

Lower right pixel X=117 Y=37

(c)

Pixel Region - newI

☑ Display pixel values 5x5

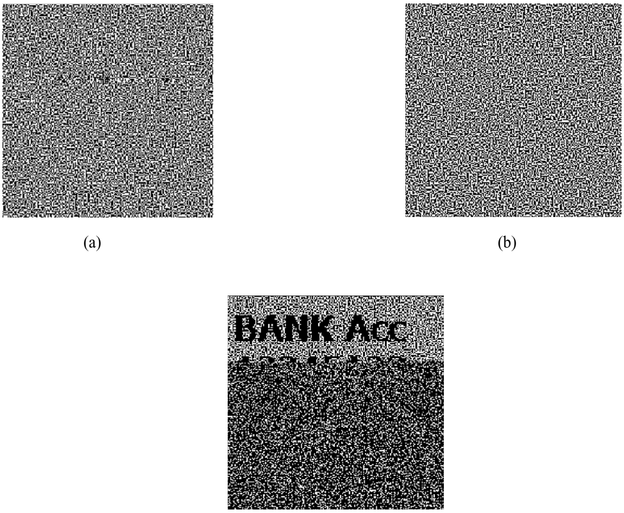
Upper left pixel X=113 Y=33

R:201 G:48 B:141	R:200 G:60 B:141	R:210 G:30 B:153	R:209 G:43 B:150	R:207 G 75 B:147
R:213 G:41 B:149	R:212 G:38 B:151	R:206 G:35 B:147	R:205 G:70 B:144	R:210 G 12 B:150
R:208 G:20 B:139	R:213 G:50 B:147	R:211 G:55 B:150	R:211 G:37 B:151	R:211 G 63 B:148
R:208 G:20 B:138	R:214 G:10 B:146	R:213 G:80 B:150	R:214 G:82 B:153	R:209 G 20 B:146
R:209 G:50 B:139	R:210 G:60 B:143	R:209 G:100 B:143	R:211 G:50 B:147	R:209 G 30 B:146

Lower right pixel X=117 Y=37

(d)

Fig. 5. Tamper detection in document authentication



(a)

(b)

(c)

Fig. 6. Extracted share and stacking reults

(Fig. 3(e) and (f)), the distribution of number of bits N_b per pixel varies greatly with different images. This is shown in Fig. 4 where distribution of bits in randomly chosen 150 pixels of four images is given.

6.1 Application in Document Authentication

With this data hiding scheme, the modification attacks against document shares in document authentication discussed in [9,10] are easily detected. Resuts obatined in this respect are given in Fig. 5. The steganographic image containing hidden share is shown in Fig. 5(a). A random 5x5 pixel region is selected in the area containing hidden information. The green plane values are changed so that the change is reflected in the stego-image. Now, this modified stego-image is used to extract the share. Small black spots can be seen in the retrieved (Fig. 6(a)) as compared to a uniform share structure as seen in Fig. 1(b)-(c). Moreover, when the two document shares are stacked together, the image that is revealed clealr shows noise from the point where modification was done to the very end (Fig.6(b)). This easily helps detect if the document was tampered with. Also, the share structure shows some visually perceptible noise.

7 Conclusion

This article proposes a new data hiding technique for EVSS schemes. It is based on a modified version of LSB substitution in which the number of bits to be hidden is determined individually for each pixel depending upon the difference of adjacent pixels. The purpose of using this scheme for hiding shares produced from a VSS scheme is to reduce the noise introduced in the cover image when data is hidden in it. Experimental results indicate that the proposed data hiding strategy gives improved PSNR values as compared to the previous scheme where noise is clearly visible in camouflage images. Moreover, the distribution of hidden bits per pixel is not uniform, as seen in simple LSB substitution. Thus, it can be concluded that afore mentioned technique improves the security of shares when hidden in cover images as compared to when they are transmitted in clear. It also prevents the modification and interception attacks discussed in [10].

References

1. Naor, M., Shamir, A.: Visual Cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)
2. Ateniese, G., Blundo, C., Santis, A.D., Stinson, D.: Extended Schemes for Visual Cryptography. Theoretical Computer Science 250(1-2), 143–161 (2001)
3. Verheul, E., Tilborg, H.V.: Constructions and Properties of k out of n Visual Secret Sharing Schemes. Designs, Codes and Cryptography 11(2), 179–196 (1997)
4. Rijmen, V., Preneel, B.: Efficient Color Visual Encryption for Shared Colors of Benetton. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070. Springer, Heidelberg (1996)
5. Lukac, R., Plataniotis, K.N.: Color Image Secret Sharing. IEE Electronic Letters 40(9), 529–530 (2004)
6. Chang, C., Tsai, C., Chen, T.: A New Scheme for Sharing Secret Color Images in Computer Network. In: Proc. International Conference on Parallel and Distributed Systems, pp. 21–27 (2000)

7. Chang, C.C., Yu, T.X.: Sharing a Secret Gray Image in Multiple Images. In: Proc. International Symposium on Cyber Worlds: Theories and Practice, pp. 230–237 (2002)
8. Sirhindi, R., Afzal, M., Murtaza, S.: An Extended Secret Sharing Scheme for Color Images with Fixed Pixel Expansion. In: Proc. International Conference on Global e-Security. CCIS, vol. 12, pp. 81–89. Springer, Heidelberg (2008)
9. Fischer, I., Herfet, T.: Visually Authenticated Communication. In: International Symposium on System and Information Security, pp. 471–474 (2006)
10. Fischer, I., Herfet, T.: Visual Document Authentication using Watermarks and Text Transformations. Journal of Computers 2(5), 44–53 (2007)
11. Chan, C.K., Cheng, L.M.: Hiding data in images by simple LSB substitution. Pattern Recognition 37, 469–474 (2004)
12. Li, S.L., Leung, K.C., Chan, C.K.: Data Hiding in Images using Adaptive LSB Substitution based on Pixel Value Differencing. In: Proc: IEEE First International Conference on Innovative Computing, Information and Control (2006)
13. Gonzalez, R.C., Woods, R.: Digital Image Fundamentals. Addison-Wesley, Reading (1992)

Efficient Multi-authorizer Accredited Symmetrically Private Information Retrieval

Mohamed Layouni[1], Maki Yoshida[2], and Shingo Okamura[2]

[1] School of Computer Science, McGill University, Montreal, Canada
[2] Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Osaka, Japan

Abstract. We consider a setting where records containing sensitive personal information are stored on a remote database managed by a *storage provider*. Each record in the database is co-owned by a fixed number of parties called data-subjects. The paper proposes a protocol that allows data-subjects to grant access to their records, to self-approved parties, without the DB manager being able to learn if and when their records are accessed. We provide constructions that allow a Receiver party to retrieve a DB record only if he has authorizations from all owners of the target record (respectively, from a subset of the owners of size greater than a threshold.) We also provide a construction where owners of the same record do not have equal ownership rights, and the record in question is retrieved using a set of authorizations consistent with a general access structure. The proposed constructions are efficient and use a pairing-based signature scheme. The presented protocol is proved secure under the Bilinear Diffie-Hellman assumption.

1 Introduction

Achieving a good quality of service and a high operational efficiency have always been a top priority for governments and businesses alike. Over the years, organizations both from the public and private sectors have experimented with a variety of technical choices and policies to improve the quality of their services. One technical choice that seems to be turning into a trend is the widespread adoption of information technologies and the continuous migration of services from the traditional paper-based world to the electronic world. The latter has a number of advantages, among which we note the greater convenience and speed to access data, which in turn translate into shorter processing delays, less errors, better statistics, higher cost-efficiency, and better auditing and fraud detection mechanisms.

Despite all the above benefits, users are still showing a certain reluctance and skepticism towards newly introduced electronic systems. The reason for this skepticism is mainly attributed to the lack of assurances about the way sensitive user data is handled, and the implications that may result from it on users' privacy.

To reduce this lack of trust, it is important that the new systems be designed in a way that gives users increased control over their data. Research on this topic

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 387–402, 2008.

received a significant attention in the past (e.g., [1,2,3,4,5,6]). More recently, a partial solution that contributes to reinforcing user's control over their data, has been proposed in [7]. This solution, called accredited symmetrically private information retrieval (ASPIR), assumes a setting where sensitive information belonging to users (data-subjects) is stored on a remote database DB managed by a party called a *Sender*. The setting includes an additional party called a *Receiver* who retrieves records from the database. The construction in [7], allows a Receiver to retrieve data owned by the user (data-subject), from a database DB managed by the Sender, such that the following three requirements are satisfied: (1) *Privacy for the data-subject:* the Receiver can retrieve a data record only if he has a valid authorization to do so from the record owner, (2) *Privacy for the Receiver:* the Sender is convinced that the Receiver's query is authorized by the owner of the target DB record, without learning any information about the content of the query, or the identity of the record owner, and (3) *Privacy for the Sender:* the Receiver cannot retrieve information about more than one record per query. For example, the Receiver cannot use an authorization from user U to learn information about database records not belonging to U.

The constructions in [7] cover a setting where each record in the database is owned by a single user. In many applications, data records are the property of several parties simultaneously rather than a single one. For example, in the healthcare domain, a medical procedure is performed by a *doctor* on a *patient* within the premises of a *hospital*. It may be natural in some jurisdictions that all three parties, namely the patient, doctor, and hospital, have a right to the database record documenting the medical procedure. As a result, a Receiver (e.g., a second doctor) who wants to have access to the above record, needs an authorization from all three record owners. With the obtained authorizations, the Receiver should be able to retrieve the target record subject to the following conditions: (1) the Receiver can retrieve the record in question only if he has the approval of all record owners, (2) the Sender is convinced that the Receiver's query is approved by the owners of the target data, without learning any information about the index of the target data, or the identity of the authorizers, and (3) the Receiver cannot retrieve information about records other than the one defined in the submitted query.

The ASPIR constructions of [7] rely on privacy-preserving digital credentials [4] to protect the anonymity of the authorizer with respect to the Sender. The digital credential primitive has been used in addition to hide the index of the retrieved record, and to guarantee the unforgeability of the issued authorizations. While highly versatile, the digital credentials of [4] do require a certain amount of computations from the different participants, especially the authorizers. In addition, the construction in [7] assumes that each record owner possesses a digital credential of the type in [4], and that he is willing to use it to issue authorizations.

In this work, we extend the ASPIR protocol of [7] to a context where each database record can have multiple owners. The protocol we present in this paper has a neater and more generic design, and uses SPIR primitives in a black-box fashion, unlike the construction in [7] which works specifically for Lipmaa's SPIR scheme [8]. Our construction is more efficient than the one in [7], and uses a lightweight pairing-based signature scheme similar to that in [9] instead of digital credentials. In this work, we also propose a t-out-of-n threshold multi-authorizer ASPIR variant, where records can be privately retrieved by a Receiver as long as he has authorizations from t out of the n owners of the target record.

The paper finally treats a setting where the owners' rights to a record are not necessarily equal. For example one could imagine a setting where an authorization from the patient is sufficient to access his medical record, while authorizations from *both* the doctor and hospital are necessary to access the same record. The latter could be useful in cases of emergency where the patient is unable to grant an authorization.

2 Related Work

The problem of managing personal data according to privacy policies defined by the data owners, has been considered by a number of authors. In [10,11], Bagga *et al.* propose a primitive called policy-based encryption. Policy-based encryption allows a user to encrypt a message with respect to an access policy formalized as a monotone Boolean expression. The encryption is such that only a user having access to a qualified set of credentials, complying with the policy, is able to successfully decrypt the message. The context in [10,11], however, is different from the one in this paper, since the goal there is to allow the user to send a secret message to a designated set of players defined by a policy. In our context, the target data is already stored in a database, and the goal is to allow parties authorized by the data owners to retrieve this data, without the database manager learning which data has been retrieved or the identity of the data owners.

In [12], Song *et al.* present a scheme allowing keyword search on encrypted data. Their setting consists of a user, and a server storing encrypted data owned by the user. The server can process search queries on the user's stored ciphertext, only if given proper authorization from the user. The scheme in [12] also supports hidden user queries, where the server conducts the search without learning anything about the content of the query. Although related to our context, it is not clear how the work in [12] can be applied to the problem we describe in this paper, since delegating querying capabilities to a third party (e.g., a Receiver) may require the user to reveal his encryption key, and thus share all of his past and future secrets. Besides, it is not clear how the scheme in [12] can hide the identity of the data-owner from the server, or how it can impose restrictions (e.g., wrt. time or usage) on the search capabilities delegated to a third party.

Finally, in [13] Aiello *et al.* consider a scenario where users privately retrieve data from a database containing a set of priced data items. The proposed protocol is called priced oblivious transfer, and allows a user U, who made an initial deposit, to buy different data items, without the database manager learning which items U is buying, subject to the condition that U's balance contains sufficient funds. We believe the construction in [13] is the first to consider imposing additional requirements on oblivious transfer protocols. While interesting in their own right, the added requirements do not address the issue of protecting the identity of the data owners.

3 Summary of Contribution and Paper Organization

We propose a multi-authorizer accredited SPIR scheme where data records stored on a Sender's database can be retrieved by a Receiver only if (1) the latter has authorizations to do so from the target record owners, and (2) without the Sender learning information about the index of the retrieved record or the identity of any of the record owners. In addition, the proposed scheme allows record owners to encode, in the issued authorizations, any privacy policy they want to enforce on their data, including the Receiver's identity, an expiry date etc. The paper also proposes a variant scheme for t-out-of-n threshold access, where a Receiver is able to retrieve a data record only if it has authorizations from at least t out of the n owners of the record. Finally, the paper treats a setting where owners of a record have unequal rights. In this setting, records are retrieved in accordance with a general access structure reflecting the non-uniformity of owners' rights.

In Section 4, we introduce few definitions, and describe the SPIR primitive which we use as a building block in our construction. In Section 5, we present our main multi-authorizer ASPIR construction. In Section 6, we evaluate the security and privacy of the proposed scheme. In Section 8, we briefly describe an extension to t-out-of-n threshold access, and treat the more general case where owners have unequal rights in Section 9. We conclude in Section 11.

4 Preliminaries

The construction we present uses a pairing-based signature scheme similar to [9], and relies on the hardness of the Bilinear Diffie-Hellman Problem (BDH). We first introduce bilinear maps, and BDH, and describe the pairing-based signature and SPIR building blocks.

Definition 1 (Admissible bilinear pairings). *Let $(\mathbb{G}_1, \times)$ and $(\mathbb{G}_2, \times)$ be multiplicative groups of the same prime order q. Assume that the discrete logarithm problem in $\mathbb{G}_1$ or $\mathbb{G}_2$ is hard, an admissible bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfying the following properties:*

- *Bilinearity: For all $P, Q \in \mathbb{G}_1$, and $\alpha, \beta \in \mathbb{Z}_q^*$, $e(P^\alpha, Q^\beta) = e(P, Q)^{\alpha\beta}$.*
- *Non-degeneracy: There exists $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1_{\mathbb{G}_2}$.*
- *Computability: Given $P, Q \in \mathbb{G}_1$, there is an efficient algorithm to compute $e(P, Q)$.*

Definition 2 (Bilinear Diffie-Hellman Problem). *Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map, and let P be a generator of $\mathbb{G}_1$. For $a, b, c \in \mathbb{Z}_q^*$, given the tuple (P, P^a, P^b, P^c) output $e(P, P)^{abc}$.*

4.1 Pairing-Based Signature Scheme

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map, and let P be a generator of $\mathbb{G}_1$. Assume the signer has a private key $sk := x \in \mathbb{Z}_q^*$, and a corresponding public key $pk := P^x$. To sign a message m, the signer computes $\sigma := H(m)^x$, where $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a public collision-resistant one-way function. The verifier accepts σ' as a valid signature on m' with respect to pk, only if $e(\sigma', P) = e(H(m'), pk)$ holds.

4.2 Symmetrically Private Information Retrieval

A private information retrieval scheme or PIR for short, involves two players: a Sender and a Receiver. The Sender manages a database DB, and answers queries on DB submitted by the Receiver. The main goal of PIR schemes is to allow the Receiver to retrieve a DB record of his choice without the Sender learning the content of his query, and without resorting to the trivial and inefficient method where the Sender just returns the whole database back to the Receiver. The property of hiding the content of the Receiver's query from the Sender is called *Privacy for the Receiver*.

PIR schemes are mainly concerned with providing *Privacy for the Receiver*. There are settings however, where the Sender too is interested in controlling access to his database. For example, the Sender could be a multimedia provider with a business model based on charging a fee for every piece of content accessed in his database. A solution to this type of settings can be obtained by using Symmetrically Private Information Retrieval schemes or SPIR for short.

A SPIR scheme allows a Receiver to efficiently retrieve records from the Sender's database such that the following two properties are assured:

- *Privacy for the Receiver*: the sender does not learn any information about the index of the target record
- *Privacy for the Sender*: the Receiver does not learn any information on the database content, other than the target record.

The above properties, namely Privacy for the Receiver, and Privacy for the Sender can be either perfect, statistical or computational. For example, Lipmaa proposes in [8] a SPIR scheme that is computationally private for the Receiver and perfectly private for the Sender.

A significant number of PIR and SPIR schemes can be found in the literature (e.g., [14,15,16,8,17]) with various performance levels, and a multitude of features such as :

- Single-DB (e.g., [15]) vs. multiple-DB Senders (e.g., [14].)
- Use of algebraic properties (e.g., homomorphic encryption [8] and ϕ-assumption [16]) vs. non-algebraic properties (e.g., existence of one-way trapdoor permutation [15].)
- Index-based (e.g., [8,16]) vs. keyword-based queries (e.g., [18].)

More information on these and other differences can be found in [19,20]. For the purpose of this paper however, we do not discuss these features any further, and use SPIR schemes in a *black-box* fashion.

Notations. In the remainder of this paper we assume that we have a SPIR scheme denoted SPIR. Let s be the secret index of the record the Receiver is interested in. The Receiver uses the public information, and possibly his private information to compute a SPIR query encoding s. We denote by $\mathsf{Q}_{\mathsf{SPIR}}$ the query the Receiver submits to the Sender. Let $\mathsf{R}_{\mathsf{SPIR}}$ be the Sender's answer to the Receiver's query. The Receiver then uses his private information and s, to recover DB$[s]$ from $\mathsf{R}_{\mathsf{SPIR}}$.

5 Protocol Description

The multi-authorizer accredited SPIR protocol we propose relies on the two building blocks described above. We start by describing a first construction in section 5.2, and then present a more efficient one in section 5.3. We assume the public parameters of the above building blocks are already known to all parties: the Sender, the Receiver, and the Authorizers.

5.1 Settings

We assume that multiple parties play the Authorizer role, as opposed to one single party as in [7]. Without loss of generality, we assume that we have *three* types of Authorizers $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$. For example, $\mathcal{A}$ could represent the Patients, $\mathcal{B}$ the Doctors, and $\mathcal{C}$ the Hospitals. In addition, our setting contains a database DB of size N managed by the Sender. Each record in DB belongs to a triplet of parties (A, B, C) from the set $\mathcal{A} \times \mathcal{B} \times \mathcal{C}$. The owners (A, B, C) of a given record may or may not have the same rights (depending on the privacy laws in place.) Section 9 treats the case where owners have unequal rights.

Next we assume that each party has an identifier *ID*, and that each record in the database is labeled with the identity of its owners, e.g., (ID_A, ID_B, ID_C). We also assume the existence of a publicly known one-to-one correspondence between *ID* triplets and the indexes of DB record, denoted $index : \mathcal{A} \times \mathcal{B} \times \mathcal{C} \rightarrow [1, N]$. Finally we assume that each DB record indexed by j, and corresponding to identity triplet $(ID_{j,1}, ID_{j,2}, ID_{j,3})$, contains a field with the owners' public keys $(pk_{j,1}, pk_{j,2}, pk_{j,3}) := (P^{x_{j,1}}, P^{x_{j,2}}, P^{x_{j,3}})$ stored in it.

5.2 First Construction

Let (A, B, C) be a tuple of owners who are willing to authorize a Receiver *RecID*, to retrieve their record indexed by $s := index(ID_A, ID_B, ID_C)$, according to a usage policy $\mathcal{P}$. Each of the owners first provides the Receiver with a signature $\sigma_i(P_m) := (P_m)^{x_i}$, for $P_m := H(s, RecID, \mathcal{P})$. Next, the Receiver prepares a SPIR query $\mathsf{Q}_{\mathsf{SPIR}}$ for index s, and submits *RecID*, $\mathcal{P}$, and $\mathsf{Q}_{\mathsf{SPIR}}$ to the Sender. Upon receiving this information, the Sender first authenticates[1] *RecID* and

[1] The receiver can be authenticated using conventional X.509 public key certificates for example. In case the identity of the receiver needs to be protected, then privacy-preserving credential systems (e.g., [4,5,6]) can be used instead.

verifies that the submitted query is compliant with usage policy $\mathcal{P}$.[2] If one of these checks fails the Sender aborts, else it proceeds with query. Next, for every Authorizer type[3], the Sender chooses a random blinding factor $\delta_i \in \mathbb{Z}_q^*$, (for the purpose of our description we have $i \in [1, 3]$.) For each record $\mathrm{DB}[j]$, the Sender computes $P_{mj} := H\left(j, \mathit{RecID}, \mathcal{P}\right)$ and $\mathrm{DB}'[j] := \mathrm{DB}[j] \times \left(\prod_{i=1}^{3} e((P_{mj})^{\delta_i}, pk_{j,i})\right)$.

The Sender then executes the SPIR scheme on $\mathsf{Q}_{\mathsf{SPIR}}$ and DB', and returns the response $\mathsf{R}_{\mathsf{SPIR}}$ to the Receiver along with $(P)^{\delta_1}$, $(P)^{\delta_2}$, and $(P)^{\delta_3}$. The Receiver first recovers $\mathrm{DB}'[s]$ from $\mathsf{R}_{\mathsf{SPIR}}$, and then computes

$$\begin{aligned}
\mathrm{DB}_0[s] &= \mathrm{DB}'[s] \ / \prod_{i=1}^{3} e\left(\sigma_i(P_m), (P)^{\delta_i}\right) \\
&= \mathrm{DB}[s] \times \prod_{i=1}^{3} e\left((P_{m,s})^{\delta_i}, pk_{s,i}\right) / \prod_{i=1}^{3} e\left((P_m)^{x_i}, (P)^{\delta_i}\right) \\
&= \mathrm{DB}[s] \times \left(\prod_{i=1}^{3} e\left((P_{m,s})^{\delta_i}, P^{x_{s,i}}\right) / e\left((P_m)^{x_i}, P^{\delta_i}\right)\right) \\
&\overset{(*)}{=} \mathrm{DB}[s] \times \left(\prod_{i=1}^{3} e\left((P_m)^{\delta_i}, P^{x_i}\right) / e\left((P_m)^{x_i}, P^{\delta_i}\right)\right) \\
&= \mathrm{DB}[s]
\end{aligned}$$

$(*)$: the equality holds because for $s = \mathit{index}(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C)$, the keys $x_{s,i}$ are no other than the secret keys x_i of owners (A, B, C). Similarly $P_{m,s} = P_m$.

In the above solution, the Sender is required to (1) make a number of pairings linear in the number of authorizer types (to compute each $e((P_{mj})^{\delta_i}, pk_{j,i})$, $i \in [1, n]$), and (2) return $(P)^{\delta_i}$ for each authorizer type. This results in computational and communication complexities linear in the number of authorizer types. We improve these complexities in the next section.

5.3 Improved Construction

Let (A, B, C) be a tuple of owners who are willing to authorize a Receiver RecID, to retrieve their record indexed by $s := \mathit{index}(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C)$, according to a usage policy $\mathcal{P}$. Each of the owners first provides the Receiver with a signature $\sigma_i(P_m) := (P_m)^{x_i}$, for $P_m := H\left(s, \mathit{RecID}, \mathcal{P}\right)$. The Receiver aggregates the σ_i's into one single signature $\mathit{Sig}(P_m) := \prod_{u \in \{A,B,C\}} \sigma_u(P_m)$. He then prepares a SPIR query $\mathsf{Q}_{\mathsf{SPIR}}$ for index s, and submits RecID, $\mathcal{P}$, and $\mathsf{Q}_{\mathsf{SPIR}}$ to the Sender as in the first construction. The Sender processes the Receiver's query as in the first construction, except that here it chooses a single random blinding factor $\delta \in \mathbb{Z}_q^*$,

[2] The policy $\mathcal{P}$ can be any Boolean statement of the form: "Receiver should be a practicing surgeon accredited by the College of Physicians **AND** Retrieval date prior to 31 July 2009" for instance. The policy can be encoded using state of the art XML format for example.

[3] As noted earlier, to keep the description simple we assumed *three* types $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$.

Receiver (*RecID*)	Public Info	Sender (Database DB)
$(P_m, \sigma_u(P_m)),\ u \in \{A,B,C\},$ for $P_m := H(s, \mathit{RecID}, \mathcal{P}),$ where $s := \mathit{index}(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C),$ and $\mathcal{P} :=$ {usage policy}		
	$\{pk_u\}_{u\in\{A,B,C\}} \triangleq \{pk_{s,i}\}_{1\le i\le 3}, e(\cdot,\cdot), P, \mathbb{G}_1 = \langle P\rangle, \mathbb{G}_2, q,$ SPIR scheme	
$\mathit{Sig}(P_m) = \prod_{u\in\{A,B,C\}} \sigma_u(P_m) = \prod_{u\in\{A,B,C\}} (P_m)^{x_u} = (P_m)^{\sum_u x_u}$		
$\mathsf{Q} = \mathsf{Q}_{\mathsf{SPIR}}(s)$	$\xrightarrow{\mathsf{Q},\ \mathit{RecID},\ \mathcal{P}}$	
		If $\mathit{RecID}, \mathcal{P}$ satisfied continue else abort Choose $\delta \in_R \mathbb{Z}_q^*$ For $j := 1$ to N do : $P_{mj} = H(j, \mathit{RecID}, \mathcal{P})$ $\mathrm{DB}'[j] = \mathrm{DB}[j] \times e\left(P_{mj}, \prod_{u=1}^{3} pk_{j,u}\right)^{\delta}$
		Execute SPIR scheme on DB' and Q
SPIR-recover $\mathrm{DB}'[s]$ from Res	$\xleftarrow{\mathsf{Res},\ P^{\delta}}$	Let $\mathsf{Res} = \mathsf{R}_{\mathsf{SPIR}}(\mathsf{Q}, \mathrm{DB}')$
Output $\mathrm{DB}_0[s] := \mathrm{DB}'[s]\,/\,e(\mathit{Sig}(P_m), P^{\delta})$		

Fig. 1. Multi-Authorizer ASPIR scheme (improved construction)

and for each $1 \le j \le N$, computes $\mathrm{DB}'[j] := \mathrm{DB}[j] \times e\left(P_{mj}, \prod_{u=1}^{3} pk_{j,u}\right)^{\delta}$. The use of a single blinding factor δ for all types of Authorizers will reduce the Sender's computational complexity from linear in the number of Authorizer types to constant. A similar reduction is achieved in the size of the Sender's response which passes from linear in the number of Authorizer types to constant.

Finally, the Sender executes the SPIR scheme on $\mathsf{Q}_{\mathsf{SPIR}}$ and DB', and returns the response $\mathsf{R}_{\mathsf{SPIR}}$ to the Receiver along with δP. The Receiver then recovers $\mathrm{DB}'[s]$ from $\mathsf{R}_{\mathsf{SPIR}}$, and computes $\mathrm{DB}_0[s] = \mathrm{DB}'[s]\,/e(\mathit{Sig}(P_m), P^{\delta})$, thereby using the aggregate signature $\mathit{Sig}(P_m)$ as if it was a "decryption key". This approach of using signatures as decryption keys is of general interest, and could be useful in the wider context of access control. A summary of the whole protocol is given in Figure 1.

It can be easily checked that $\mathrm{DB}_0[s]$ computed by the Receiver is the desired record $\mathrm{DB}[s]$.

$$\begin{aligned}
\mathrm{DB}_0[s] &= \mathrm{DB}'[s]\,/\,e(\mathit{Sig}(P_m), P^{\delta})\\
&= \mathrm{DB}[s] \times e(P_m, \prod_{u=1}^{3} pk_u)^{\delta}\,/\,e((P_m)^{\sum_{u=1}^{3} x_u}, P^{\delta})\\
&= \mathrm{DB}[s] \times e(P_m, P^{\sum_{u=1}^{3} x_u})^{\delta}\,/\,e((P_m)^{\sum_{u=1}^{3} x_u}, P)^{\delta}\\
&= \mathrm{DB}[s]
\end{aligned}$$

Remark. The usage policy $\mathcal{P}$ encoded in P_m can be any privacy policy the owners want enforced on their record. This may include usage limitations such as an expiry date, a description of what is considered an acceptable usage scenario etc. Note that by binding authorizations to a specific Receiver exclusively, the protocol is able to prevent pooling attacks[4].

6 Security and Privacy Evaluation

Definition 3 (Valid Authorization). *Let (A, B, C) be the owners of a record in the Sender's DB, indexed by $s = index(ID_A, ID_B, ID_C)$. For a given usage policy $\mathcal{P}$, a Receiver is said to have a valid authorization under $\mathcal{P}$, from owner $O \in \{A, B, C\}$, if and only if the Receiver has a valid signature from O on $P_m = H(s, ReceiverID, \mathcal{P})$, and $\mathcal{P}$ is satisfied at the time the authorization is used.*

Definition 4 (Secure ASPIR protocols). *An ASPIR protocol is said to be secure if (1) the protocol satisfies the "privacy for Receiver" and "privacy for Sender" properties usually provided by conventional SPIR schemes, and (2) a Receiver cannot retrieve a given record with non-negligible probability unless he has authorizations from all owners of that record. For the special cases of threshold ASPIR (resp., ASPIR with unequal ownership rights), we require the Receiver to have authorizations from a subset of the owners of size greater than a threshold (resp., a subset that is part of a given access structure.)*

Theorem 1. *Assuming the Bilinear Diffie-Hellman problem is hard and the* SPIR *primitive secure, the protocol of Figure 1 is a secure ASPIR protocol.*

Proof. The protocol of Figure 1 is by assumption based on a secure SPIR primitive. By examining the exchange of messages, it is easy to see that the protocol of Figure 1 satisfies the "privacy for Receiver" and "privacy for Sender" properties already provided by the underlying $SPIR$ primitive. In the following we examine the second security criterion of definition 4.

We show that if an Adversary $\mathcal{A}_{ASPIR}$ can retrieve a record that $\mathcal{A}_{ASPIR}$ is not authorized to obtain then the Bilinear Diffie-Hellman problem can be solved. In other words, we show how to construct an Adversary $\mathcal{A}_{BDH}$ that uses $\mathcal{A}_{ASPIR}$ to solve the Bilinear Diffie-Hellman problem.

Let s be the index of the record targeted by the Adversary $\mathcal{A}_{ASPIR}$ playing the role of a malicious Receiver. Let (ID_A, ID_B, ID_C) be the identity tuple of the corresponding owners, i.e., $s = index(ID_A, ID_B, ID_C)$. The Adversary $\mathcal{A}_{ASPIR}$ submits a query and retrieves record DB$[s]$ from the Sender's response without having all required authorizations from owners tuple (A, B, C). In the absence of authorizations from owners tuple (A, B, C), the best scenario for the adversary is to have valid signatures from two (out of the three) owners. Without loss of generality, assume he has signatures from A and B.

[4] Pooling attacks occur when different receivers combine their authorizations in order to gain access to records they were not able to get access to, each on his/her own.

For any given instance $(P', (P')^a, (P')^b, (P')^c)$ of the BDH problem, the Adversary $\mathcal{A}_{BDH}$ obtains $(abc) \cdot e(P', P')$ by interacting with $\mathcal{A}_{ASPIR}$ and playing the role of the owners A and B, and the Sender as follows.

1. $\mathcal{A}_{BDH}$ chooses random elements x_A, x_B of $\mathbb{Z}_q^*$ and sets $P = P'$, $pk_A = (P')^{x_A}$, $pk_B = (P')^{x_B}$, and $pk_C = (P')^c$.
2. $\mathcal{A}_{BDH}$ gives P and $\{pk_i\}_{i \in \{A,B,C\}}$ to $\mathcal{A}_{ASPIR}$.
3. $\mathcal{A}_{BDH}$ sets $P_m = (P')^b$ for the parameters s, *RecID*, and $\mathcal{P}$ (the hash function H is assumed as a random oracle in this proof).
4. $\mathcal{A}_{BDH}$ computes signatures $\sigma_A(P_m) = (P_m)^{x_A}$ and $\sigma_B(P_m) = (P_m)^{x_B}$, and gives them to $\mathcal{A}_{ASPIR}$, along with s, *RecID*, and usage policy $\mathcal{P}$.
5. $\mathcal{A}_{ASPIR}$ submits $\mathsf{Q} := \mathsf{Q}_{\mathsf{SPIR}}(s)$, *RecID*, and usage policy $\mathcal{P}$ to $\mathcal{A}_{BDH}$.
6. $\mathcal{A}_{BDH}$ sets :
   - $\mathrm{DB}_0[j] := e\left(P_m, ((P')^a)^{(x_A+x_B)}\right)$ for all j.
   - $P^\delta := (P')^a$

   $\mathcal{A}_{BDH}$ then executes SPIR on DB_0 and Q and returns $\mathsf{Res} = \mathsf{R}_{\mathsf{SPIR}} = \mathsf{SPIR}(\mathrm{DB}_0, \mathsf{Q})$ and P^δ to $\mathcal{A}_{ASPIR}$.
7. $\mathcal{A}_{ASPIR}$ computes (this step could be done earlier)
$$Sig(P_m) := \prod_{i \in \{A,B,C\}} \sigma_i(P_m) := (P')^{b(x_A+x_B+c)}$$
8. $\mathcal{A}_{ASPIR}$ recovers $\mathrm{DB}_0 = \mathrm{DB}_0[s]$ from Res and computes
$$\begin{aligned}\mathrm{DB} &= \mathrm{DB}_0 \,/\, e(Sig(P_m), P^\delta)\\ &= e\left(P_m, (P')^{a(x_A+x_B)}\right) \,/\, e((P')^{b(x_A+x_B+c)}, (P')^a)\\ &= e\left((P')^b, (P')^{a(x_A+x_B)}\right) \,/\, e((P')^{b(x_A+x_B+c)}, (P')^a)\\ &= e\,(P', P')^{ab(x_A+x_B)} \,/\, e(P', P')^{ab(x_A+x_B+c)}\\ &= e\,(P', P')^{(ab(x_A+x_B)-ab(x_A+x_B+c))}\\ &= e\,(P', P')^{-(abc)}\end{aligned}$$
9. $\mathcal{A}_{BDH}$ outputs $\mathrm{DB}^{-1} = e\,(P', P')^{abc}$

$\mathcal{A}_{BDH}$ can solve the BDH problem using $\mathcal{A}_{ASPIR}$. Therefore, assuming the BDH problem is hard, computing a record without all the required valid authorizations is unfeasible. ∎

The above proof can be straightforwardly generalized to the case where records belong to n owners, for n arbitrary. Similar theorems can be proved for the protocol variants of Sections 8, and 9.

7 Performance Analysis

In this analysis we focus mainly on exponentiation operations; group operations such as multiplications are significantly cheaper. A pairing operation can be reduced to a single exponentiation of size less than the group order (as noted in [21]), and is therefore considered as a small-size exponentiation.

It is worth noting at this point that all SPIR schemes require $\Omega(|\mathrm{DB}|)$ computations form the Sender; if this is not the case, then the Sender will not touch at least one record in the database, and thus can safely infer that the untouched records are not being sought in the Receiver's query, thereby violating the Receiver's privacy. As a result of this observation, the Sender's overall computations cannot be expected to drop below this linear lower bound.

Let n be the number of owners of each record in the Sender's database, and let N be the database size. In addition to the basic operations required by the underlying SPIR scheme, our protocol requires : (a) each owner of the target record to perform one pre-computable exponentiation in $\mathbb{G}_1$, (b) the Receiver to perform a pre-computable n-point multiplication in $\mathbb{G}_1$ (to compute $Sig(P_m)$), and one pairing, and (c) the Sender to perform N exponentiations and N pairings. Despite the increase in functionalities, the protocol we propose does not lead to higher computational cost compared to that of the underlying SPIR scheme (which is linear in N.) Similarly, our communication performance is equivalent to that of the underlying SPIR scheme, since we increase the amount of exchanged data only by a small constant. This is negligible, since the best known communication complexity for SPIR achieved so far is $\mathcal{O}\big(\log^2(N)\big)$ [17,8].

8 Extension to Threshold Access

In some applications it may be useful to provide a mechanism to allow a Receiver to privately recover a certain record as long as he has authorizations from t out of the n record owners. As in the basic case, the Sender should not learn the identity of the Authorizers or the index of the retrieved record. We do this using ideas similar to those in [22]. In the following, we only point out the changes from the basic protocol of section 5.

Assume the record owners jointly select a master secret key $MSK := x \in \mathbb{Z}_q^*$, and distribute it verifiably among themselves in a (t, n)-secret sharing scheme. We note that there is no need for a third party in the secret sharing procedure. The n record owners can generate secret key MSK, and privately distribute the shares among themselves without help from a trusted third party using protocols such as [23,24]. The secret generation is such that no shareholder knows MSK individually. Due to space limitations we do not expose the details of those schemes here. Let $x_u,\ u \in [1, n]$ be the n secret shares, and $(sk_u, pk_u) := (x_u, P^{x_u}),\ u \in [1, n]$ the private/public key pairs of the record owners. The master secret key x can be written as a Lagrange interpolation of any subset of shares x_u, of size greater or equal to t. Let $MPK := P^x$ be the corresponding master public key. Finally we assume that each DB record indexed by j, and corresponding to identity triplet $(ID_{j,1}, \cdots, ID_{j,n})$, contains a field with the master public key MPK_j stored in it. Note that given the owners' public keys $(pk_{j,1}, \cdots, pk_{j,n})$, anyone can reconstruct the corresponding master public key MPK_j by simple Lagrange interpolation.

A Receiver holding authorizations $(P_m, \sigma_u(P_m))$ from at least t record owners $\{u_1, \cdots, u_t\}$, can reconstruct a signature on P_m with respect to the master public key MPK, by computing $Sig(P_m) = \prod_{v=1}^{t} \sigma_{u_v}(P_m)^{L_{u_v}} = (P_m)^{\sum_{v=1}^{t} L_{u_v} x_{u_v}} = (P_m)^x$, where L_{u_v} denote the appropriate Lagrange coefficients[5]. The Receiver then proceeds with the protocol as in the basic case, and submits $\mathsf{Q_{SPIR}}$, RecID, and usage policy $\mathcal{P}$ to the Sender.

The Sender checks the consistency of the submitted query with the Receiver's identity and usage setting, and chooses a random blinding factor $\delta \in \mathbb{Z}_q^*$. For each record in the database indexed by j, the Sender computes P_{mj}, and $\mathrm{DB}_0[j] = \mathrm{DB}[j] \times e\left(P_{mj}, MPK_j\right)^{\delta}$. The rest of the protocol is similar to the one in Section 5.

9 Extension to Authorizers with Unequal Rights

Up to this point we have assumed that the owners of a given record all have equal rights. In other words, if a record belongs to (A, B, C) then an authorization from A is worth exactly the same as one from B or C. In some settings however, owners of a record do not have equal rights. For instance in the healthcare context, a medical record belonging to (patient A, doctor B, hospital C) should be accessible only if authorizations are provided, say from A alone, or B and C together. Authorizations from B or C alone are not sufficient. More generally, for a record R owned by a set $O = \{A_1, \cdots, A_n\}$, we denote by $\mathcal{A} \subset 2^O$ the subsets of O whose authorizations are sufficient to access R. The set $\mathcal{A}$ is called a generalized access structure. In the following we show how secret sharing with a generalized access structure [25] can be used to realize multi-authorizer ASPIR in a context where owners have unequal rights to their record.

Consider a database record R, and assume R's owners agree on a generalized access structure $\mathcal{A}$. Using a method similar to that of Section 8, R's owners jointly select a master secret key $MSK := x \in \mathbb{Z}_q^*$, and split it into shares among themselves, according to the access structure $\mathcal{A}$. The secret generation and distribution are such that no shareholder knows MSK individually, and no help from a secret sharing dealer is needed. More details on how this is done are given in the example below. Each owner ends up with a share of information on MSK, that he uses as a signing key. The master public key MPK corresponding to MSK is stored in a field within record R, as in the threshold construction of Section 8. A Receiver then obtains signatures from a subset of owners as in the threshold case. Next, the Receiver combines the partial signatures using Lagrange interpolation in order to recover a valid signature with respect to master key MPK. Recovering this signature is possible only if the Receiver obtains partial signatures from a set of owners that is part of the access structure $\mathcal{A}$.

[5] The values of the L_{u_v}'s depend only on the values of the u_v's.

Example. Let R be a record belonging to (A_1, A_2, A_3, A_4), who agree on access structure $\mathcal{A} = \{\{A_1, A_2, A_3\}, \{A_1, A_4\}, \{A_2, A_4\}, \{A_3, A_4\}\}$. Let $x \in \mathbb{Z}_q^*$ be the master secret key *MSK* that (A_1, A_2, A_3, A_4) select jointly. Let (x_1, x_2, x_3, x_4) be shares of x in a $(4, 4)$-threshold secret sharing scheme. Assume we have a mechanism to securely distribute share tuples (x_2, x_4) to A_1, (x_3, x_4) to A_2, (x_1, x_4) to A_3, and (x_1, x_2, x_3) to A_4. It can be easily seen that the distributed share tuples do satisfy the access structure $\mathcal{A}$. Further details on how share tuples are determined in the general case, can be found in [25].

The received x_i's are used by the owners as private signing keys to issue authorizations. For example, a Receiver authorized by $\{A_1, A_4\} \in \mathcal{A}$, obtains $(P_m, \sigma_2(P_m), \sigma_4(P_m))$ from A_1, and $(P_m, \sigma_1(P_m), \sigma_2(P_m), \sigma_3(P_m))$ from A_4, where $\sigma_i(P_m) = (P_m)^{x_i}$ for $1 \leq i \leq 4$. The Receiver then computes the signature on P_m with respect to master key *MPK*, by interpolating the σ_i's as follows : $\sigma(P_m) = \prod_{v=1}^{4} (\sigma_v(P_m))^{L_v}$, where L_v denote the appropriate Lagrange coefficients. The reconstructed signature is later used by the Receiver to "decrypt" $\mathrm{DB}'[s]$ as in the original ASPIR protocol of Section 5. The rest of the protocol remains the same as in the threshold case.

Now we give a brief overview on how the master secret x is jointly selected by (A_1, A_2, A_3, A_4), and how the shares are generated and distributed. For $1 \leq i \leq 4$, owner A_i chooses $s_i \in_R \mathbb{Z}_q^*$, and generates a random 3^{rd}-degree polynomial in $\mathbb{Z}_q$, $f_i(X) = s_i + \sum_{j=1}^{3} a_{ij} X^j$. Let $f(X) = \sum_{i=1}^{4} f_i(X)$. If we set $x = \sum_{i=1}^{4} s_i$, then $\{x_j = f(j),\ 1 \leq j \leq 4\}$ is valid set of (4,4)-threshold shares of x. Note that x is uniquely determined at this point, and yet unknown to any of the A_i's individually. Next, the share tuples are distributed as follows. Consider for instance the share tuple (x_2, x_4) intended for A_1. For $2 \leq i \leq 4$, owner A_i sends $(f_i(2), f_i(4))$ to A_1. Next, A_1 obtains the desired shares by computing $x_j = \sum_{i=1}^{4} f_i(j)$, for $j \in \{2, 4\}$. The remaining share tuples for A_2, A_3, and A_4 are distributed in the same way. The share distribution above can be made verifiable using the technique of [23].

10 The Case of an Owner Tuple Possessing Multiple Records

So far, we have assumed that each tuple (A, B, C) could own at most one single record. In the following we briefly discuss the case where a tuple of owners may possess $k \geq 1$ records. The goal now is to allow these owners to issue an authorization to the Receiver so that he can retrieve their k records. One trivial way to do this is as follows. First, add one argument to the $\mathit{index}(\ldots)$ function, specifying the rank of record. For example, $s_i = \mathit{index}(A, B, C, i)$ will now denote the index of the i^{th} record (among k) belonging to (A, B, C). The owners now give the Receiver an authorization for each $\mathrm{DB}[s_i]$, and the retrieval proceeds as in the basic case.

To avoid the issuing of multiple authorizations, we can use the following method. The value of P_m in the authorization issued to the Receiver is now computed as $P_m = H(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C, \mathit{RecID}, \mathcal{P})$, and each of the owners provides the

Receiver with a signature $\sigma_i(P_m) := (P_m)^{x_i}$. The Receiver then aggregates the σ_i's into one single signature $Sig(P_m) := \prod_{u\in\{A,B,C\}}\sigma_u(P_m)$ as in section 5.3. A similar modification is required on the Sender's side as well. For $j \in [1, N]$, the Sender computes the P_{mj}'s rather as $P_{mj} = H(ID_{j,1}, ID_{j,2}, ID_{j,3}, RecID, \mathcal{P})$. Note that the identities $ID_{j,u}$, $u \in [1, 3]$, of the record owners are readily available to the Sender along with the corresponding public keys $pk_{j,u}$, $u \in [1, 3]$. The Sender then computes $\mathrm{DB}'[j]$ from $\mathrm{DB}[j]$ as in section 5.3 using the new value of P_{mj} instead. As a result of the above modifications, we note that for all indexes $s_i = index(ID_{j,1}, ID_{j,2}, ID_{j,3}, i)$, $i \in [1, k]$, referencing the records belonging owner tuple $(ID_{j,1}, ID_{j,2}, ID_{j,3})$, the value of P_{mj} is the same, and the entries $\mathrm{DB}[s_i]$ are all encrypted with the same "key": $e(P_{mj}, \prod_{u=1}^{3} pk_{j,u})^{\delta}$. The Receiver finally SPIR retrieves the entries $\mathrm{DB}'[s_i]$ one by one, and decrypts them using his aggregate signature $Sig(P_m)$ as in the basic case.

In the above scheme, the Receiver SPIR retrieves the the $\mathrm{DB}'[s_i]$'s separately. This can be improved using a method based on the hybrid encryption paradigm [26]. First we modify the setting to include two databases DB_1 and DB_2. Each entry in DB_1 is used to store a key corresponding to a triplet of owners. The database DB_2 on the other hand, is used to store the actual owners' records encrypted under the keys kept in DB_1, using some data encapsulation mechanism (DEM)[6][26]. DB_2 is such that the records belonging to a given tuple of owners are all encrypted under the *same* key. In order to grant access to their records, the owners (A, B, C) give the Receiver an authorization to retrieve their encryption key from DB_1 (using the construction of section 5.3.) And using this key, the Receiver decrypts all the DB_2 records belonging to (A, B, C). Note that if DB_2 can be made public, the Receiver does not need to run the SPIR scheme again to retrieve the encrypted records.

11 Conclusion

The paper presents a special access control protocol for databases containing sensitive personal data. In particular, the described constructions allow a Receiver to retrieve a record in the database, if and only if (a) he has authorizations from all (resp. a threshold portion of) the target record owners, and (b) the context in which the database is queried, is consistent with a usage policy chosen by the owners of the target record, and embedded in authorizations issued to the Receiver. The above is achieved without the database manager being able to learn any information about the index of the target record or the identity of its owners. The proposed construction is proved secure under the BDH assumption. The paper also presents a construction where the owners of a record do not have equal ownership rights. The protocol we propose in this paper is more efficient than the one in [7] and can be constructed with any SPIR primitive. Despite the increase in functionality, the presented protocol does not lead to a complexity higher than that of the underlying SPIR.

[6] Note that DEM could be any symmetric-key encryption scheme (e.g., AES.)

Acknowledgments. The first author thanks the IWT-SBO project (ADAPID) "Advanced Applications for Electronic Identity Cards in Flanders", and the Universitary Mission of Tunisia in North America for their support.

References

1. Golle, P., McSherry, F., Mironov, I.: Data collection with self-enforcing privacy. In: ACM Conference on Computer and Communications Security, pp. 69–78 (2006)
2. Ateniese, G., de Medeiros, B.: Anonymous e-prescriptions. In: WPES, pp. 19–31 (2002)
3. Yang, Y., Han, X., Bao, F., Deng, R.H.: A smart-card-enabled privacy preserving e-prescription system. IEEE Transactions on Information Technology in Biomedicine 8(1), 47–58 (2004)
4. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
5. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
6. 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)
7. Layouni, M.: Accredited symmetrically private information retrieval. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 262–277. Springer, Heidelberg (2007)
8. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
9. 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)
10. Bagga, W., Molva, R.: Policy-based cryptography and applications. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 72–87. Springer, Heidelberg (2005)
11. Bagga, W., Molva, R.: Collusion-free policy-based encryption. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 233–245. Springer, Heidelberg (2006)
12. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society, Los Alamitos (2000)
13. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
14. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
15. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373 (1997)
16. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)

17. Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005)
18. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003 (1998)
19. Ostrovsky, R., Skeith III, W.E.: A survey of single-database private information retrieval: Techniques and applications. In: Public Key Cryptography, pp. 393–411 (2007)
20. Gasarch, W.I.: A survey on private information retrieval (column: Computational complexity). Bulletin of the European Association for Theoretical Computer Science 82, 72–107 (2004)
21. Boyen, X.: A promenade through the new cryptography of bilinear pairings. In: IEEE Information Theory Workshop—ITW 2006, pp. 19–23. IEEE Press, Los Alamitos (2006)
22. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1991)
24. Ingemarsson, I., Simmons, G.J.: A protocol to set up shared secret schemes without the assistance of a mutually trusted party. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 266–282. Springer, Heidelberg (1991)
25. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. Electronics and Communications in Japan (Part III: Fundamental Electronic Science) 72(9), 56–64 (1989)
26. Shoup, V.: A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report2001/112 (2001), http://eprint.iacr.org/

Specification of Electronic Voting Protocol Properties Using ADM Logic: FOO Case Study

Mehdi Talbi[1,2], Benjamin Morin[1], Valérie Viet Triem Tong[1], Adel Bouhoula[2], and Mohamed Mejri[3]

[1] Supélec, Équipe SSIR (EA 4039), Avenue de la Boulaie, Cesson-Sévigné, France
firstname.lastname@supelec.fr

[2] Digital Security Unit, Higher School of Communication, Tunis, Tunisia
bouhoula@planet.tn

[3] LSFM Research Group, Computer Science Department, Université Laval, Sainte-Foy, Qc., Canada
momej@ift.ulaval.ca

Abstract. It is a well known fact that only formal methods can provide a proof that a given system meets its requirements. For critical systems (e.g. nuclear reactors, aircraft), the use of these methods becomes mandatory. Electronic voting is also one of these critical systems since the stakes are important: democracy. In this context, we propose in this paper, the use of the ADM logic in order to specify security properties (fairness, eligibility, individual verifiability and universal verifiability) of electronic voting protocols. These properties are first specified in a general form, and then adapted to the FOO protocol as a case study. Our goal is to verify these properties against a trace-based model. The choice of the ADM logic is motivated by the fact that it offers several features that are useful for trace analysis. Moreover, the logic is endowed with a tableau-based proof system that leads to a local model checking which enables an efficient implementation.

1 Introduction

The multiple benefits of e-voting (electronic voting) push many governments to change their traditional methods and to move toward the e-democracy. In fact, electronic voting allows, amongst others, to reduce the cost of the vote, to get the results faster and more accurately, to improve the accessibility and to reduce the risk of human errors. Over the last years, several e-voting systems have been proposed and studied. In particular, there is a strong interest in Internet voting systems since they make the vote more convenient and there is a hope that they could increase the participation rate. Like e-commerce, Internet voting is based on cryptographic protocols, however it must meet several and complicated security properties (fairness, eligibility, anonymity, receipt-freeness, individual and universal verifiability) making them more challenging than electronic commerce applications. Some of these properties can even seem contradictory at first sight. For example, the voter must be able to verify that his vote was really counted

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 403–418, 2008.

(individual verifiability), and at the same time cannot prove to a third party (coercer) that he voted as agreed (receipt-freeness).

In the literature, we distinguishes two schemes of voting protocols: those based on blind signatures [1,2,3], and those based on homomorphic encryption [4,5]. These techniques attempt in two different ways to reach the properties of electronic voting protocols. However, the complexity of these properties makes it difficult to prove that an e-voting protocol respects even a part of them.

The main intent of this work is to show how to use ADM logic [6], special variant of the μ-calculus modal logic [7], in order to specify four properties of e-voting protocols: fairness, eligibility, individual and universal verifiability. Our specification is dedicated to voting protocols based on blind signatures. First, the properties will be specified in a general form, and then they will be adapted to the FOO protocol [1] as a case study. Our intention is to verify these properties against a trace-based model. More precisely, we want to check the satisfaction relation $t \models \phi$, where t is a finite trace representing a valid execution of the analyzed protocol, and ϕ an ADM formula characterizing a property of the protocol in question.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 gives an informal description of the FOO protocol. Section 4 introduces some notations used to specify a protocol and presents the model of the FOO protocol. Section 5 presents the ADM logic. Section 6 gives the specification of electronic voting properties in the ADM logic. Section 7 introduces the tableau-based proof system of ADM, and gives a concrete example showing how it can be used to check properties against the FOO model. Finally, some concluding remarks on this work and future research are ultimately sketched as a conclusion in Section 8.

2 Related Work

The literature is rich in works dealing with formal verification of security protocols. However, there are only few formal works [8,9,10,11,12,13,14] related to electronic voting protocols. This is mainly due to their lack of maturity compared to other ones such as key distribution or authentication protocols, and to the complexity of the involved techniques. Indeed, they involve advanced cryptographic primitives (e.g. bit commitment, blind signature), and rely on complex channels (e.g. anonymous, private).

Process algebra and model-checking remain the main formal techniques used for the verification of electronic voting protocols. In these approaches, the protocol is basically specified using the most suitable process algebra (e.g. ACP [15], applied π-calculus [16]), and properties are specified using the most appropriate logic (e.g. CTL [17], μ-calculus), or defined in terms of observational equivalence. For instance, Kremer and Ryan propose in [10,13] to use the applied π-calculus in order to verify four properties (eligibility, fairness, anonymity and receipt-freeness) of electronic voting protocols. The analyzed protocol (FOO in [13]) is specified first in the applied π-calculus, and the properties are then specified, generally, in terms of observational equivalence. The anonymity property, for example, is stated as follows: the protocol FOO satisfies the anonymity property if the process where two voters vote v_1 and v_2 (respectively), is observationally equivalent to the process where the two voters swap their votes. In [14], Mauw et al. use the ACP process algebra to specify the FOO protocol, and propose a method to measure the anonymity property.

Knowledge-based logics have been also used in [8,11,12] to formally analyze e-voting protocol. For instance, in [11] we can find a formalization of anonymity property using an epistemic logic. This property is expressed, with regard to a simplified version of the FOO protocol (specified as Kripke structure), using the PDL logic [18]. This logic offers, among others, the possibility to write properties allowing to reason about the knowledge of an agent a of the system (e.g. Alice) with respect to a proposition p (e.g. "Bob voted Yes").

In our work, we use a trace-based model to represent a protocol since traces constitute the simplest and most natural execution model considered so far in the literature. Also, many automatic trace generators are available in the literature, and they can be adapted for Internet voting protocols. Moreover, a security flaw in a cryptographic protocol is usually illustrated by a trace that violates a security property. We use the ADM logic to specify electronic voting properties, as it offers several features that are useful for trace analysis (linearity, recursion, temporal modalities). It was used in [6] to specify classical security properties such as secrecy and authentication, but also to specify electronic commerce properties such as good atomicity and money atomicity. This last property aims to ensure that the number of $credit()$ actions in the trace is equal to the number of $debit()$ actions. This feature (counting actions), which is not possible with other logics such as μ-calculus, is helpful to specify the universal verifiability property for example. Indeed, in such a property one has to verify if the number of occurrences of an action a ($cast(v)$) is equal to the number of occurrences of an action b ($publish(v)$). In this paper, we focus on the specification of four electronic voting properties: fairness, eligibility, individual and universal properties. Anonymity and receipt-freeness properties will be considered in a later paper.

3 FOO Protocol

We will take the FOO protocol proposed by Fujioka et al. in [1] as a case study, in order to validate our specification on a concrete example, but also to complete the formal analysis made by previous works [8,9,11,13,14], notably by the specification of the universal verifiability property. We give here a brief informal description of the protocol.

The protocol takes place in three phases: **registering**, **voting** and **results publishing**. Principals involved in the protocol are: Voters $V_i \in L_{voter}$, $i \in \{1 \dots n\}$ (L_{voter} represents the list of eligible voters), an administrator A, and a collector C. The role of authority A in the protocol is to provide tokens (blind signature) for legitimate voters. The role of collector C is to collect the votes and to publish election results. Let (pka, ska) be the public/private key pair associated with A, and (pkv_i, skv_i) the public/private key pair associated with the voter V_i. Let L_{vote} be the list of possible votes. For example, $L_{vote} = \{yes, no\}$ in the case of a referendum. The variable $v_i \in L_{vote}$ represents the vote of V_i.

The FOO protocol starts by a **registration phase**. A voter $V_i \in L_{voter}$ chooses his vote v_i, computes the commitment $\{v_i\}_{r_i}$ using a random key r_i. Then, he blinds the obtained value using the key b_i (generated randomly), and finally digitally signs the blinded committed vote using his private key skv_i, and sends the signed message to the administrator A, together with his identity. A checks if the voter V_i is an eligible voter (i.e. $V_i \in L_{voter}$), has not registered yet, and that the received signature is valid. If all these verifications succeed, then A registers the voter V_i, "blindly" signs the blinded committed vote sent by the voter using his secret key ska, and sends this signature to the voter V_i. V_i then, verifies the signature of A, and if this test succeeds, "unblinds" the signature received from A to obtain a committed vote digitally signed by A. This signed message represents the token allowing the voter to cast his vote during the second phase of the protocol (i.e. voting phase).

The voting phase starts after a fixed deadline (e.g. all voters have registered). The signed committed vote obtained during the first phase is sent anonymously to the collector C, which checks the validity of the signature. If this test succeeds, C registers V's vote (i.e. $(\{v_i\}_{r_i}, \{\{v_i\}_{r_i}\}_{ska})$) as element $elem_i$ of the list of registered votes.

The last phase of the protocol (**Results Publishing**) starts after a fixed deadline (e.g. reception of all encrypted votes). The collector C publishes the list of all encrypted votes (i.e. $(\{v_i\}_{r_i}, \{\{v_i\}_{r_i}\}_{ska}, elem_i)$, $i \in \{1, \ldots, p_1\}, p_1 \leqslant |L_{voter}|$). The voter V_i can verify that his vote is in the list published by C. If it is the case, V_i sends anonymously the key r_i^{-1} together with element $elem_i$ identifying his vote. The Collector C decrypts the i-th element of the list of encrypted votes, verifies the validity of V_i's vote $(v_i \in L_{vote})$, and registers the vote of V_i (i.e. $\{v_i\}_{r_i}, \{\{v_i\}_{r_i}\}_{ska}, r_i^{-1}, v_i)$. At the end of this stage (i.e. after a fixed deadline), collector C publishes the list of correctly opened votes (i.e. $(\{v_i\}_{r_i}, \{\{v_i\}_{r_i}\}_{ska}, r_i^{-1}, v_i)$, $i = \{1, \ldots, p_2\}$, $p_2 \leqslant |L_{voter}|$).

Based on the description given above, we give in the next section the model of the FOO protocol.

4 Model

Our intention is to specify electronic voting properties, and then to check them against a protocol model. In this section, we give some notations for protocol specification and use them to model the FOO protocol.

4.1 Notations for Protocol Specification

An electronic voting protocol can be viewed as a particular example of cryptographic protocols. They consist on communication steps (send and receive) and computation steps (e.g. checking the signature of a message) allowing entities to exchange messages, through communication channels (e.g. public, anonymous), in order to achieve a given goal (e.g. authentication, sharing a secret, voting). In order to verify our properties, it is fundamental to have a simple representation of protocols that captures all the information needed to perform their analysis. This representation comprises the format of exchanged messages, the structure of external and internal actions, and the representation of communication channels. In this paper, we will use notations proposed in [6] where a protocol P is defined as a finite sequence of statements of the form given by Table 1.

Table 1. Protocol Representation

$$\begin{pmatrix} \cdots\cdots \\ i - act(p_1, ..., p_n) \\ \cdots\cdots \\ i - A \rhd B : m \\ i - B \lhd A : m \\ \cdots\cdots \\ i - act(q_1, ..., q_m) \\ \cdots\cdots \end{pmatrix}$$

Table 2. Message Syntax

$$m ::= M \mid v \mid (m, m') \mid \{m\}_k \mid k$$

The statement given in Table 1 is the representation of the execution of a protocol P at a step i. More precisely, an execution step of a protocol is represented as an execution of an external action (send or receive), optionally preceded and followed by a set of internal actions. An internal action is a computation step. It is represented in the form $i\text{-}act(p_1, ..., p_n)$ to denote execution of the action act over the set of parameters $\{p_1, ..., p_n\}$. An external action is a communication step. It takes the form $i\text{-}A \rhd B : m$ (resp. $i\text{-}A \lhd B : m$) to denote the fact that A sends to (resp. receives from) B the massage m at step i. Format of messages is given by the BNF-grammar given by Table 2. We use capital letter to denote principals (e.g. V_i represents the voter i). v is a variable (e.g. vote). (m, m') is used to denote a composed message. Finally, $\{m\}_k$ represents an encrypted message m with the key k. A key can be symmetric ($k \in Sym\_Keys$), private ($k \in Sec\_Keys$), public ($k \in Pub\_Keys$) or a blinder key ($k \in Bln\_Keys$). Although "blinding" is similar to symmetric key encryption, we introduce the set Bln_Keys in order to model some properties of blinded signed messages. Thus, $\{m\}_k$ represents committed message as well as blinded messages and signed messages. We assume that each signed message $\{m\}_{sk}$ is sent with message m.

Finally, in our specification, a channel is by default anonymous. Information sent through anonymous channels can be intercepted by an intruder, but cannot be traced back to the sender. This characteristic can be achieved by use of Mix Nets introduced by Chaum in [19]. If the channel is non anonymous (public channel), then we will add the identity of the sender to the message sent.

4.2 FOO Protocol Model

The adopted model is a trace-based model, where a trace is a sequence of actions (external and internal actions) representing a valid execution of the protocol. A trace is said valid if all the messages sent by the intruder can be derived from the intruder's cumulated knowledge, and all the involved principals respect the protocol. The set of traces $\mathcal{T}$ is defined as follows:

$$\begin{cases} \epsilon \in \mathcal{T} & \text{(empty trace)} \\ \text{If } \tau \in \mathcal{T} \text{ and a is an action, then } \tau.a \in \mathcal{T} & \text{(concatenation)} \end{cases}$$

A protocol can be modelled as a subset of traces $\mathcal{P}$ of all possible traces $\mathcal{T}$. The set $\mathcal{P}$ is built up using the same formalism as that proposed by Paulson in [20] where a trace $t \in \mathcal{P}$, resulting from the execution of a given protocol, is built up inductively through a set of inference rules ($\frac{premise}{conclusion}$). These rules are interpreted as follows: if the conditions of the premise are fulfilled, then the conclusion will be executed. Table 3 gives the set of rules related to the FOO protocol. The rule *init* is relative to the initialisation of the protocol. This means that we start first with an empty trace. Rules $V_{request\_token}$, V_{cast_vote} and V_{send_key} model the role of voters in the protocol. For example, the rule V_{cast_vote} means that if the trace contains the reception of blinded signed message from the authority A, if this signature is valid (i.e. the side condition A_{sign} holds), and if the voter V respects the deadlines, then the trace can be extended with the sending of the vote of V to collector C. Rules $A_{register\_voter}$ and $A_{provide\_token}$ model the role of authority A in the protocol. Similarly, rules $C_{save\_encrypted\_vote}$, $C_{publish\_encrypted\_vote}$, C_{save_vote}, and $C_{publish\_vote}$ model the role of collector C in the protocol. We distinguish three deadlines in the FOO protocol. These deadlines are modelled by rules $Deadline_i$ $(i = 0, 1, 2)$ which mean that if we reach a certain time limit denoted by T_i $(i = 0, 1, 2)$, then the trace can be extended with the internal action (j-$Deadline(T_i)$). Deadline at T_0 marks the end of the registration phase. Deadline at T_1 marks the end of the voting phase. Finally, deadline at T_2 marks the end of opening ballots phase. A special care must be taken when modelling "deadlines". For instance, in rule $Deadline_0$, before extending the trace with (3-$Deadline(T_0)$), we have to ensure first that rule $A_{provide\_token}$ cannot be executed. This captures the fact that authority A must provide a token (i.e. blind signature) for all voters that have registered before the deadline at T_0. The rule *Receive* states that a principal can get a message only if it has been previously sent to her. The rule *Broadcast* models message broadcasting to all voters (i.e. the receiver '*' represents the set of all voters: $L_{voter}(P)$). Finally, the rule *Intruder* models the capacity of the intruder to send a message built up from its cumulated knowledge.

When modelling cryptographic protocols we assume the presence of an intruder that can overhear messages, intercept messages, replay messages and generate new messages using his cumulated knowledge. Deriving new messages is captured by the Dolev-Yao model [21] which defines some basic rules such as *compose*, *decompose*, *decrypt* and *encrypt*. For instance, the rule *encrypt* states that if m (message) and k (key) are known to intruder, then he can deduce $\{m\}_k$. In e-voting protocols based on blind signature, we have to add an additional rule allowing to extract a signed message from a blinded signed message:

$$unblind\_sig \; \frac{b^{-1} \in M_{\Downarrow} \quad \{\{m\}_b\}_{sk} \in M_{\Downarrow}}{\{m\}_{sk}} \; b \in Bln\_Keys \;\; sk \in Sec\_Keys$$

With $M_{\Downarrow}$: Set of messages known to the intruder.

Table 3. FOO Protocol Model

$Init$	$\overline{\epsilon \in \mathcal{P}}$
$V_{request\_token}$	$\dfrac{t \in \mathcal{P} \quad (3 - Deadline(T_0)) \notin \bar{t}}{t.(1 - V \rhd A : ((V, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{skv})) \in \mathcal{P}}$
$A_{register\_voter}$	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (1 - Save(V)) \notin \bar{t} \\ (1 - A \lhd V : ((V, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{skv})) \in \bar{t} \\ (3 - Deadline(T_0)) \notin \bar{t} \end{cases}}{t.(1 - Save(V)) \in \mathcal{P}} \; V \in L_{voter}(P) \;\; V_{sign}$
$A_{provide\_token}$	$\dfrac{t \in \mathcal{P} \quad (1 - Save(V)) \in \bar{t} \quad (2 - A \rhd V : ((A, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{ska})) \notin \bar{t}}{t.(2 - A \rhd V : ((A, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{ska})) \in \mathcal{P}}$
$Deadline_0$	$\dfrac{t \in \mathcal{P} \quad \bar{A}_{provide\_token}(t) \quad (3 - Deadline(T_0)) \notin \bar{t}}{t.(3 - Deadline(T_0)) \in \mathcal{P}}$
V_{cast_vote}	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (2 - V \lhd A : ((A, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{ska})) \in \bar{t} \\ (3 - Deadline(T_0)) \in \bar{t} \quad (4 - Deadline(T_1)) \notin \bar{t} \end{cases}}{t.(3 - V \rhd C : \{v\}_r, \{\{v\}_r\}_{ska}) \in \mathcal{P}} \; A_{sign}$
$C_{save\_encrypted\_vote}$	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (3 - C \lhd V : (\{v\}_r, \{\{v\}_r\}_{ska})) \in \bar{t} \quad (3 - Deadline(T_0)) \in \bar{t} \\ (3 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, elem)) \notin \bar{t} \quad (4 - Deadline(T_1)) \notin \bar{t} \end{cases}}{t.(3 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, elem)) \in \mathcal{P}} \; A_{sign}$
$Deadline_1$	$\dfrac{t \in \mathcal{P} \quad (3 - Deadline(T_0)) \in \bar{t} \quad (4 - Deadline(T_1)) \notin \bar{t}}{t.(4 - Deadline(T_1)) \in \mathcal{P}}$
$C_{publish\_encrypted\_vote}$	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (3 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, elem)) \in \bar{t} \\ (4 - Deadline(T_1)) \in \bar{t} \quad (4 - C \rhd * : (((C, \{v\}_r), \{\{v\}_r\}_{ska}), elem)) \notin \bar{t} \end{cases}}{t.(4 - C \rhd * : (((C, \{v\}_r), \{\{v\}_r\}_{ska}), elem)) \in \mathcal{P}}$
V_{send_key}	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (3 - V \rhd C : (\{v\}_r, \{\{v\}_r\}_{ska})) \in \bar{t} \quad (4 - Deadline(T_1)) \in \bar{t} \\ (4 - V \lhd C : (((C, \{v\}_r), \{\{v\}_r\}_{ska}), elem)) \in \bar{t} \quad (6 - Deadline(T_2)) \notin \bar{t} \end{cases}}{t.(5 - V \rhd C : (r^{-1}, elem)) \in \mathcal{P}}$
C_{save_vote}	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (3 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, elem)) \in \bar{t} \\ (4 - Deadline(T_1)) \in \bar{t} \quad (5 - C \lhd V : (r^{-1}, elem)) \in \bar{t} \\ (5 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, r^{-1}, v)) \notin \bar{t} \\ (6 - Deadline(T_2)) \notin \bar{t} \end{cases}}{t.(5 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, r^{-1}, v)) \in \mathcal{P}} \; v \in L_{vote}(P)$
$Deadline_2$	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (4 - Deadline(T_1)) \in \bar{t} \quad (6 - Deadline(T_2)) \notin \bar{t} \\ \bar{C}_{publish\_encrypted\_vote}(t) \end{cases}}{t.(6 - Deadline(T_2)) \in \mathcal{P}}$
$C_{publish\_vote}$	$\dfrac{\begin{cases} t \in \mathcal{P} \quad (5 - Save(\{v\}_r, \{\{v\}_r\}_{ska}, r^{-1}, v)) \in \bar{t} \\ (6 - C \rhd * : ((((C, \{v\}_r), \{\{v\}_r\}_{ska}), r^{-1}), v)) \notin \bar{t} \\ (6 - Deadline(T_2)) \in \bar{t} \end{cases}}{t.(6 - C \rhd * : ((((C, \{v\}_r), \{\{v\}_r\}_{ska}), r^{-1}), v)) \in \mathcal{P}}$
$Receive$	$\dfrac{t \in \mathcal{P} \quad (i - X \rhd Y : m) \in \bar{t}}{t.(i - Y \lhd X : m) \in \mathcal{P}}$
$Broadcast$	$\dfrac{t \in \mathcal{P} \quad (i - X \rhd * : m) \in \bar{t}}{t.(i - V_1 \lhd X : m) \ldots (i - V_n \lhd X : m) \in \mathcal{P}}$
$Intruder$	$\dfrac{t \in \mathcal{P} \quad m \in Message(t)_{\Downarrow}}{t.(i - Y \lhd X : m) \in \mathcal{P}}$

Message(t): Set of messages in the trace t
Message(t)$_{\Downarrow}$: Set of messages deduced by the intruder from trace t
$\bar{t}$: Set of components in the trace t
$\bar{R}$(t): Conditions of the premise of rule R are not fulfilled in the trace t

In Table 4, we give an example of a valid trace built up inductively using the rules given in Table 3. This trace is obtained by applying successively the rules $Init$, $V_{request\_token}$, $Receive$, $A_{register\_voter}$, $A_{provode\_token}$, $Receive$, $Deadline_0$, $Intruder$, ... We will show in Section 6.4 that this trace, which respects the specification of the protocol, actually violates the universal verifiability property.

Table 4. Trace Violating the Universal Verifiability Property

$$\text{t} = \begin{cases} 1\text{-}V \rhd A : ((V, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{skv}) \\ 1\text{-}A \lhd V : ((V, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{skv}) \\ 1\text{-}Save(V) \\ 2\text{-}A \rhd V : ((A, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{ska}) \\ 2\text{-}V \lhd A : ((A, \{\{v\}_r\}_b), \{\{\{v\}_r\}_b\}_{ska}) \\ 3\text{-}Deadline(T_0) \\ 3\text{-}C \lhd V : (\{v_a\}_{r_a}, \{\{v_a\}_{r_a}\}_{ska}) \\ 3\text{-}Save(\{v_a\}_{r_a}, \{\{v_a\}_{r_a}\}_{ska}, elem_a) \\ 4\text{-}Deadline(T_1) \\ 4\text{-}C \rhd * : (((C, \{v_a\}_{r_a}), \{\{v_a\}_{r_a}\}_{ska}), elem_a) \\ 5\text{-}C \lhd V : (r_a^{-1}, elem_a) \\ 5\text{-}Save(\{v_a\}_{r_a}, \{\{v_a\}_{r_a}\}_{ska}, r_a^{-1}, v_a) \\ 6\text{-}Deadline(T_2) \\ 6\text{-}C \rhd * : ((((C, \{v_a\}_{r_a}), \{\{v_a\}_{r_a}\}_{ska}), r_a^{-1}), v_a) \end{cases}$$

5 ADM Logic

The ADM logic can be viewed as a special variant of μ-calculus. Although it has initially been designed for the specification of electronic commerce properties, it is also very appropriate for the electronic voting issue. The logic includes modalities such as "necessity", "possibility", "before", "after". Moreover, it allows to consume resources (linearity) and to specify recursive formulae. The syntax of the ADM logic is based on patterns. A pattern is an abstraction of a trace, where some actions are replaced by variables. The syntax of a pattern is defined by the BNF-grammar given in Table 5. According to this syntax, a pattern is a sequence of actions and pattern variables. ϵ stands for empty pattern, x is a pattern variable and a is an action, which itself may contain variables. For example, in pattern $p = x_1.(x_s - X_v \rhd A : x_m).x_2$, we identify action variables (x_1 and x_2), a step variable (x_s), a principal variable (X_v), and a message variable (x_m).

The syntax of the logic is defined by the BNF-grammar given in Table 6. The symbols $\neg$ and $\wedge$ represent, respectively, negation and conjunction. X is a formula variable. $\nu X.\phi$ is a recursive formula. The greatest fixed point operator ν binds all free occurrences of X in ϕ. There is a syntactic restriction on the body of $\nu X.\phi$ which stipulates that any occurrence of X in ϕ must occur under the scope of an even number of negations. Finally, $[p_1 \looparrowright p_2]$ is a modal operator indexed by the two patterns p_1 and p_2. It should be noted here, that the set of pattern variables in p_2 must be included in the set of pattern variables in p_1 (no new variables appearing in p_2). A trace t satisfies the formula $[p_1 \looparrowright p_2]\phi$, if for all substitutions σ such that $p_1\sigma = t$, the new trace $t_1 = p_2\sigma$ satisfies the remaining part of the formula (ϕ). In this respect, $[p_1 \looparrowright p_2]\phi$ has two effects. First, the part p_1 allows us to verify if something has happened somewhere in the trace t. Second, the part p_2 allows us to modify the trace (delete some actions, substitute some actions by others, add some actions) in such a way that the remainder of the formula (ϕ) will be verified on the modified version of the trace described by p_2.

Notice that usual shortcuts such that $\vee$, $\rightarrow$, $\leftrightarrow$ can be used with their usual meaning. Note also, that from the modal operator $[-]$, we can derive the modal

Table 5. Pattern Syntax

$p ::= \epsilon \mid x.p \mid a.p$

Table 6. ADM Logic Syntax

$\phi ::= X \mid \neg\phi \mid [p_1 \looparrowright p_2]\phi \mid \phi_1 \wedge \phi_2 \mid \nu X.\phi$

operator $\langle - \rangle$ ($\langle p_1 \looparrowright p_2\rangle\phi \equiv \neg[p_1 \looparrowright p_2]\neg\phi$). The formula $\langle p_1 \looparrowright p_2\rangle\phi$ is interpreted as follows: verify if there exists at least one substitution σ that makes equal the analyzed trace t to the pattern p_1, and the rest of the formula (ϕ) has to be satisfied in the new version of the trace defined by $p_2\sigma$. Similarly, from the greatest fixed point ν, we can derive the tautology formula tt ($tt \equiv \nu X.X$) and the least fixed point μ ($\mu X.\phi \equiv \neg\nu X.\neg\phi[\neg X/X]$). $\phi[\Gamma/X]$ represents the simultaneous replacement of all free occurrences of X in ϕ by Γ.

Examples of formulae given in [6] allow to better understand the ADM logic constructs.

6 Properties Specification

In this section, we will use the ADM logic to specify four electronic voting properties. This specification is relative to blind signature schemes [1,2,3]. Protocols based on such schemes proceed in several steps. In a first phase, the voter contacts the authority managing the election in order to obtain a signature on a message which is generated beforehand by the voter. This signature affixed by the authority constitutes the token which will allow the voter to cast his vote. Hence, only legitimate voters (those which have obtained a token) can vote.

Before specifying properties, we give in Table 7 rewriting rules allowing to simplify the specification of our properties. The idea behind these rules is to preprocess the trace in order to reason on decrypted and atomic messages. Rules of Table 7 allow us to derive from the trace t its normal form $t_{\downarrow}$. For example, the normal form $t_{\downarrow_1}$, which is obtained by applying repetitively rule R_1, is the trace where all composed messages (x_{m_1}, x_{m_2}) in t are decomposed into x_{m_1} and x_{m_2} messages. Transformations made by rules R_2 and R_3, capture the fact that if the trace contains the message $\{x_m\}_{x_k}$ followed/preceded by a message x_k^{-1}, then the message $\{x_m\}_{x_k}$ will be replaced by x_m. The only one difference between R_2 and R_3 is the order of messages $\{x_m\}_{x_k}$ and x_k^{-1} in the trace. Hence, the normal form $t_{\downarrow_{123}}$ is the trace where all composed messages (x_{m_1}, x_{m_2}) are decomposed into x_{m_1} and x_{m_2} messages, and all encrypted messages $\{x_m\}_{x_k}$ are replaced by x_m messages, if the corresponding key x_k^{-1} appear in the trace. Note that the rewriting system is convergent, and hence the resulting normal form is unique.

For a better readability of properties, constant terms will be written in boldface. This will allow to avoid confusion between pattern variables and constants in formulae. Now we are ready to specify our properties. They are firstly formulated generally, then applied in the particular case of the FOO protocol.

6.1 Fairness

This property implies that the protocol should not allow the disclosure of partial results. Such results can influence the vote of the voters not having voted yet.

Table 7. Decomposition and Decryption Rewriting Rules

$$\boxed{\begin{aligned}
R_1 :\ & x.(x_i\text{-}X_s \rhd X_r : (x_{m_1}, x_{m_2})).y \rightarrow \\
& x.(x_i\text{-}X_s \rhd X_r : x_{m_1}).(x_i\text{-}X_s \rhd X_r : x_{m_2}).y \\[1em]
R_2 :\ & x.(x_{i_1}\text{-}X_{s_1} \rhd X_{r_1} : \{x_m\}_{x_k}).y.(x_{i_2}\text{-}X_{s_2} \rhd X_{r_2} : x_k^{-1}).z \rightarrow \\
& x.(x_{i_1}\text{-}X_{s_1} \rhd X_{r_1} : x_m).y.(x_{i_2}\text{-}X_{s_2} \rhd X_{r_2} : x_k^{-1}).z \\[1em]
R_3 :\ & x.(x_{i_1}\text{-}X_{s_1} \rhd X_{r_1} : x_k^{-1}).y.(x_{i_2}\text{-}X_{s_2} \rhd X_{r_2} : \{x_m\}_{x_k}).z \rightarrow \\
& x.(x_{i_1}\text{-}X_{s_1} \rhd X_{r_1} : x_k^{-1}).y.(x_{i_2}\text{-}X_{s_2} \rhd X_{r_2} : x_m).z
\end{aligned}}$$

The specification of this property is given by the formula $\mathcal{F}_P(L_{vote}, T)$ which means that if a vote $v \in L_{vote}$ can be deduced by an observer, then this deduction occurs after a special event denoted $d(T)$, and if we remove one occurrence of v, then the remaining trace must still satisfy the whole formula. Intuitively, a vote v can be deduced, if it appears in the normal form of the trace given by $t_{\downarrow_{123}}$ (i.e. decomposition and decryption of messages in t). Hence, the trace t (i.e. trace to be analyzed) satisfies the fairness property if the formula $\mathcal{F}_P$ is satisfied by the trace $t_{\downarrow_{123}}$.

$$\begin{aligned}
\mathcal{F}_P(L_{vote}, T) \equiv \nu X.(\textstyle\bigwedge_{\boldsymbol{v \in L_{vote}}} & (\langle x.(x_i\text{-}X_s \rhd X_r : \boldsymbol{v}).y \looparrowright \epsilon\rangle tt \rightarrow \\
& \langle x.\boldsymbol{d(T)}.y.(x_i\text{-}X_s \rhd X_r : \boldsymbol{v}).z \looparrowright x.\boldsymbol{d(T)}.y.z\rangle X))
\end{aligned}$$

The fairness property corresponding to the FOO protocol is obtained by replacing $d(T)$ with the internal action (4-*Deadline*(T_1)), since it represents the beginning of the opening ballots phase.

6.2 Eligibility

This property means that only eligible voters can vote, and any legitimate voter can vote only once during an election. We express this property in terms of a subset of formulae.

The first of these formulae expresses the fact that the voting protocol must allow each legitimate voter to vote. This property is usually referred to as democracy. In the ADM logic, this property can be expressed according to the formula given hereafter:

$$\mathcal{D}_P(L_{voter}, A, k, T) \equiv \textstyle\bigwedge_{\boldsymbol{V \in L_{voter}}} (\langle x.a.actp \looparrowright \epsilon\rangle \rightarrow \langle x.a.y.b.z.\boldsymbol{d(T)}.w \looparrowright \epsilon\rangle tt)$$

$$With \begin{cases} a = \boldsymbol{request}(\boldsymbol{V}, \boldsymbol{A}, x_m) \\ b = \boldsymbol{token}(\boldsymbol{A}, \boldsymbol{V}, x_m, \boldsymbol{k}) \\ actp = z_1.(x_i\text{-}\boldsymbol{act_1}(\boldsymbol{V}, x_m)).z_2 \ldots z_p.(x_i\text{-}\boldsymbol{act_p}(\boldsymbol{V}, x_m)).z_{p+1}) \end{cases}$$

This property means that if the trace contains an action $request(V, A, x_m)$ (request for signature, on message x_m, from a voter $V \in L_{voter}$ to an authority A), and if certain conditions (denoted as internal actions $act_i(V, x_m), i \in \{1 \ldots p\}$) hold on the pair (V, x_m) (e.g. valid signature), then the trace must contain an action $token(A, V, x_m, k)$ (representing the sending of the message $\{x_m\}_k$ from

A to V). This action must occur before the deadline at T. Message $\{x_m\}_k$ represents the message x_m, blindly signed with the key k. More precisely, $\{x_m\}_k$ represents the token allowing the voter to cast his vote later in the next phase of the protocol.

The democracy property corresponding to the FOO protocol is obtained by the following substitutions:

$$With \begin{cases} a = (\mathbf{1}\text{-}\boldsymbol{V} \rhd \boldsymbol{A} : \{\{x_v\}_{x_r}\}_{x_b}) \\ b = (\mathbf{2}\text{-}\boldsymbol{A} \rhd \boldsymbol{V} : \{\{\{x_v\}_{x_r}\}_{x_b}\}_{\boldsymbol{ska}}) \\ d(T) = (\mathbf{4}\text{-}\boldsymbol{Deadline(T_1)}) \\ actp = z_1.(\mathbf{1}\text{-}\boldsymbol{Save(V)}).z_2 \end{cases}$$

Action a represents the request for signature. Action b represents the token sent by A. This action must occur before the deadline at T_1 (i.e. action $d(T)$). This means that A must provide a token for each legitimate voter before the ending of the voting phase. Finally, 1-$Save(V)$ represents the only condition to obtain a token. Indeed, in our modelling, actions that are specific to the validity of signatures are not reported in the trace. The same goes for the legitimacy of the voters. However, for the sake of convenience, we grouped these actions together, and represented them with only one action ($Save(V)$), obtained by the rule $A_{register\_voter}$. A trace t satisfies the democracy property, if $\mathcal{D}_P$ is satisfied by the normal form of the trace given by $t_{\downarrow_1}$ (i.e. decomposition of messages in the trace t).

The eligibility property implies also that only legitimate voters can vote, and thus only eligible voters are allowed to gain a token. This property can be formalized in a similar way to the previously specified property (by the opposite reasoning):

$$\mathcal{K}_P(A,k) \equiv \nu X.((\langle x.b.y \looparrowright \epsilon\rangle tt \rightarrow \langle x.a.actp.b.y \looparrowright x.actp^{-}.y\rangle X)$$

$$With \begin{cases} a = request(X_v, \boldsymbol{A}, x_m) \\ b = token(\boldsymbol{A}, X_v, x_m, \boldsymbol{k}) \\ actp = z_1.(x_i\text{-}\boldsymbol{act_1}(X_v, x_m)).z_2 \ldots z_p.(x_i\text{-}\boldsymbol{act_p}(X_v, x_m)).z_{p+1}) \\ actp^{-} = z_1.z_2 \ldots z_p.z_{p+1} \end{cases}$$

This property is interpreted in the following way: if the trace contains a token sent by an authority A (action b), then the trace must contain a request for this token (action a), coming from a legitimate voter (actions $act_i(X_v, m)$, $i \in \{1 \ldots p\}$ appear in the trace), and if we remove one occurrence of these actions (a, b and $act_i(X_v, m)$, $i \in \{1 \ldots p\}$), then the trace must still satisfy the formula.

Finally, the eligibility property implies that a legitimate voter can vote only once. In the case of a blind signature scheme, a voter can vote several times in two manner: by replaying his vote, or by obtaining several tokens from the administrator. Generally, this can be checked by the formula given hereafter, and which expresses the fact that an action $a \in \{a_1, ..., a_n\}$ does not appear more than once in the trace.

$$\mathcal{I}_P(\{a_1, ..., a_n\}) \equiv \neg(\bigvee_{\boldsymbol{a \in \{a_1, ..., a_n\}}}(\langle x.\boldsymbol{a}.y.\boldsymbol{a}.z \looparrowright \epsilon\rangle tt))$$

The version of the formula $\mathcal{I}_P$ corresponding to the FOO protocol is given hereafter:

$$\neg(\langle x.a.y.a.z \looparrowright \epsilon\rangle tt \vee \langle x.b.y.b'.z \looparrowright \epsilon\rangle tt)$$

$$With \begin{cases} a = (x_i\text{-}\boldsymbol{C} \rhd * : \{x_v\}_{x_r}) \\ b = (\mathbf{2}\text{-}\boldsymbol{A} \rhd X_v : \{\{x_v\}_{x_r}\}_{x_b} \\ b' = (\mathbf{2}\text{-}\boldsymbol{A} \rhd X_v : \{\{x_v\}_{x'_r}\}_{x'_b} \end{cases}$$

Action a represents the publication of encrypted votes. Thus, the first part of the formula expresses the ability of the protocol to reject double votes. Actions b and b' are sending operations in which the authority A sends a token to the same voter. Thus, the second part of the formula expresses the fact that each legitimate voter is authorized to obtain only one token. A given trace t of the FOO protocol satisfies the property $\mathcal{I}_P$, if this formula is satisfied by the normal form of the trace given by $t_{\downarrow_1}$.

The eligibility property is given by the conjunction of the three properties given above:

$$\mathcal{E}_P \equiv \mathcal{D}_P \wedge \mathcal{K}_P \wedge \mathcal{I}_P$$

6.3 Individual Verifiability

This property implies that the protocol allows the voter to verify that his vote was really counted. We formalize this property in the following way: if the trace t contains an action $cast(V, v, k_1, ..., k_m)$ (vote of the voter $V \in L_{voter}$ encrypted with a set of keys $\{k_1, ...k_m\}$), and if this vote is accepted by an authority C (valid vote, valid signature, ... represented as internal actions $act_i()$, $i \in \{1 \ldots p\}$), then the trace must contain an action $publish(C, v, k_1, ..., k_m)$ executed by the authority C. We specify this property (formula $\mathcal{VI}_P(L_{voter}, C)$) in the same manner as the democracy property defined previously. Actually, the individual verifiability property supplements, in a certain sens, the democracy property.

$$\mathcal{VI}_P(L_{voter}, C) \equiv \bigwedge\nolimits_{\boldsymbol{V \in L_{voter}}}(\langle x.a.actp \looparrowright \epsilon\rangle tt \rightarrow \langle x.a.y.b.z \looparrowright \epsilon\rangle tt)$$

$$With \begin{cases} a = cast(\boldsymbol{V}, x_v, x_{k_1}, ..., x_{k_m}) \\ b = publish(\boldsymbol{C}, x_v, x_{k_1}, ..., x_{k_m}) \\ actp = z_1.(x_i\text{-}\boldsymbol{act_1()}).z_2 \ldots z_p.(x_i\text{-}\boldsymbol{act_p()}).z_{p+1}) \end{cases}$$

The individual verifiability property corresponding to the FOO protocol is given hereafter:

$$\mathcal{VI}_P(L_{voter}, C) \equiv \bigwedge\nolimits_{\boldsymbol{V \in L_{voter}}}(\langle x.a_1.actp_1.a_2.actp_2 \looparrowright \epsilon\rangle tt \rightarrow \langle x.a_1.y.a_2.z.b.w \looparrowright \epsilon\rangle tt)$$

$$With \begin{cases} a_1 = (\mathbf{3}\text{-}\boldsymbol{V} \rhd \boldsymbol{C} : (\{x_v\}_{x_r}, \{\{x_v\}_{x_r}\}_{\boldsymbol{ska}})) \\ a_2 = (\mathbf{5}\text{-}\boldsymbol{V} \rhd \boldsymbol{C} : (x_r^{-1}, x_{elem})) \\ b = (\mathbf{6}\text{-}\boldsymbol{C} \rhd * : ((((\boldsymbol{C}, \{x_v\}_{x_r}), \{\{x_v\}_{x_r}\}_{\boldsymbol{ska}}), x_r^{-1}, x_v))) \\ actp_1 = y_1.(\mathbf{3}\text{-}\boldsymbol{Save}(\{x_v\}_{x_r}, \{\{x_v\}_{x_r}\}_{\boldsymbol{ska}}, x_{elem}).y_2 \\ actp_2 = y_3.(\mathbf{5}\text{-}\boldsymbol{Save}(\{x_v\}_{x_r}, \{\{x_v\}_{x_r}\}_{\boldsymbol{ska}}, x_r^{-1}, x_v)).y_4 \end{cases}$$

In the FOO voting scheme, action *cast*() is realized in two steps: action a_1 represents the first step (V sends his vote encrypted with r) and action a_2 represents the second step (V sends the key r allowing to open his ballot). Internal actions in pattern variables $actp_1$ and $actp_2$ must also occur in the trace. They denote, that V's vote is valid. If all these conditions are satisfied, then the trace must contain an action *publish*() (action b) which corresponds to the publication of V's vote.

6.4 Universal Verifiability

The Universal verifiability is a property which guarantees that any voter can be convinced of the validity of the published votes. This property can be formalized in a similar way to the previously specified property (by the opposite reasoning). Thus, for each published vote, the trace must contain a vote submitted by an eligible voter. The formula given hereafter represents the specification of the universal verifiability property in the ADM logic:

$$\mathcal{VU}_P(L_{voter}, C) \equiv \nu X.(((\langle x.b.y \looparrowright \epsilon\rangle \mathit{tt} \rightarrow \bigvee_{\boldsymbol{V \in L_{voter}}} (\langle x.a.y.b.z \looparrowright x.y.z\rangle)X))$$

$$With \begin{cases} a = \boldsymbol{cast}(\boldsymbol{V}, x_v, x_{k_1}, ..., x_{k_m}) \\ b = \boldsymbol{publish}(\boldsymbol{C}, x_v, x_{k_1}, ..., x_{k_m}) \end{cases}$$

This property is interpreted in the following way: if the trace contains an action *publish*() executed by an authority C, then the trace must contain an action *cast*() executed by a voter V, and if we remove one occurrence of these actions, then the remaining trace must still satisfy the formula.

The universal verifiability property corresponding to the FOO protocol is given hereafter:

$$\mathcal{VU}_P(L_{voter}, C) \equiv \nu X.(((\langle x.b.y \looparrowright \epsilon\rangle \mathit{tt} \rightarrow \bigvee_{\boldsymbol{V \in L_{voter}}} (\langle x.a_1.y.a_2.z.b.w \looparrowright x.y.z.w\rangle)X))$$

Actions a_1, a_2 and b in $\mathcal{VU}_P$ are those defined previously in the individual verifiability property.

Note that the FOO protocol does not satisfy the universal verifiability property. Indeed, if the voter V abstains from voting after registering, then the authority A can add its own vote instead of V's vote. The trace t corresponding to this scenario is given by Table 4. In this case, it is clear that the trace t does not satisfy the universal verifiability property. Indeed, t contains a published vote $(\{v_a\}_{r_a}, \{\{v_a\}_{r_a}\}_{ska}, r_a^{-1}, v_a)$ which does not correspond to any vote submitted by an eligible voter V (the trace t does not contain the participation of a voter V having voted $(\{v_a\}_{r_a}, \{\{v_a\}_{r_a}\}_{ska})$ during the voting phase).

7 Formal Verification of Properties

In this section we describe briefly the tableau-based proof system, and show, through a concrete example, how we can use it to verify if a trace of the FOO protocol satisfies or not a given formula.

Table 8. Tableau-Based Proof System

$R_\neg$	$\dfrac{H, b, e, \sigma \vdash t \in \neg\phi}{H, \neg b, e, \sigma \vdash t \in \phi}$	
$R_\wedge$	$\dfrac{H, b, e, \sigma \vdash t \in (\phi_1 \wedge \phi_2)}{H, b_1, e, \sigma \vdash t \in \phi_1 \quad H, b_2, e, \sigma \vdash t \in \phi_2}$	$b_1 \times b_2 = b$
R_ν	$\dfrac{H, b, e, \sigma \vdash t \in \nu X.\phi}{H[X \mapsto H(X) \cup t], b, e, \sigma \vdash t \in \phi[\nu X.\phi/X]}$	$t \notin H(X)$
$R_{[]}$	$\dfrac{H, b, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi}{H, b_1, e, \sigma_1 \circ \sigma \vdash p_2\sigma\sigma_1 \in \phi \ \ldots \ H, b_n, e, \sigma_n \circ \sigma \vdash p_2\sigma\sigma_n \in \phi}$	C
With	$C = \begin{pmatrix} \{\sigma_1, \ldots, \sigma_n\} = \{\sigma' \vert p_1\sigma\sigma' = t\} \neq \emptyset \\ and \\ b_1 \times \ldots\ldots\ldots\ldots\ldots \times b_n = b, n > 0 \end{pmatrix}$	$\begin{pmatrix} \epsilon \times \epsilon = \epsilon \\ \epsilon \times \neg = \neg \\ \neg \times \neg = \neg \end{pmatrix}$

Table 9. Successful Leafs and Unsuccessful Leafs

Successful Leaf	Unsuccessful Leaf
$\theta = (H, \epsilon, e, \sigma \vdash t \in X)$ and $t \in e(X)$	$\theta = (H, \epsilon, e, \sigma \vdash t \in X)$ and $t \notin e(X)$
$\theta = (H, \neg, e, \sigma \vdash t \in X$ and $t \notin e(X)$	$\theta = (H, \neg, e, \sigma \vdash t \in X$ and $t \in e(X)$
$\theta = (H, \epsilon, e, \sigma \vdash t \in \nu X.\phi)$ and $t \in H(X)$	$\theta = (H, \neg, e, \sigma \vdash t \in \nu X.\phi)$ and $t \in H(X)$
$\theta = (H, \epsilon, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi)$ and $\{\sigma' \vert p_1\sigma\sigma' = t\} = \emptyset$	$\theta = (H, \neg, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi)$ and $\{\sigma' \vert p_1\sigma\sigma' = t\} = \emptyset$

The ADM logic is endowed with a tableau-based proof system that has been proved to be sound and complete in [6] with respect to the denotational semantics. In fact, this tableau-based semantics leads to a local model checking. Rules defined in Table 8 allows to capture in a deductive way whether a trace t satisfies a formula ϕ or not. The proof rules operate on sequents of the form $H, b, e, \sigma \vdash t \in \phi$. H is a mapping in $[\mathcal{V} \rightarrow 2^{\mathcal{T}}]$, where $\mathcal{V}$ is the set of variables and $\mathcal{T}$ the set of traces. A special care should be devoted to the rules that handle recursion. Mapping H is used to this end. The flag b is a variable in $\{\epsilon, \neg\}$. It is used to remember if we are dealing with the formula ϕ or the formula $\neg\phi$. Environment e is a mapping in $[\mathcal{V} \rightarrow 2^{\mathcal{T}}]$. It is used to give a semantic to the formula X and to deal with recursive formulae. Finally, σ is a substitution, t a trace, and ϕ a formula.

A sequent θ has a successful tableau if there exists a finite tableau having θ as a root and all its leaves are successful. A leaf θ is successful when it meets one of the conditions given in the first column of Table 9. It is unsuccessful when it meets one of the conditions given in the second column.

Rules of Table 8 are interpreted as follows:

- $R_\neg$: verifying if a trace t satisfies or not a formula $\neg\phi$, amounts to verify if t satisfies or not the formula ϕ and then to decide for the formula $\neg\phi$. Flag b is used to this end.
- $R_\wedge$: proving that the sequent $H, b, e, \sigma \vdash t \in (\phi_1 \wedge \phi_2)$ has a successful tableau, amounts to verify that the sequent $H, b_1, e, \sigma \vdash t \in \phi_1$ and the sequent $H, b_2, e, \ \sigma \vdash t \in \phi_2$ have successful tableaux for some b_1 and b_2 such that $b = b_1 \times b_2$.
- R_ν: proving that the sequent $H, b, e, \sigma \vdash t \in \nu X.\phi$ has a successful tableau, amounts to prove that the sequent $H[X \mapsto H(X) \cup t], b, e, \sigma \vdash t \in \phi[\nu X.\phi/X]$ has a successful tableau. Moreover, the condition $t \notin H(X)$ must hold.

- $R_{[]}$: this rule means that if $\{\sigma_1 \dots \sigma_n\} = \{\sigma' | p_1\sigma\sigma' = t\}$ and for all $i \in \{1 \dots n\}$, $p_2\sigma\sigma_i$ satisfies ϕ, then we deduce that t satisfies ϕ.

Now, suppose that we want to verify if the trace t of the protocol FOO given in Table 4 is satisfied or not by the formula ϕ given hereafter:

$$\phi \equiv \neg(\langle x.b.y.b'.z \looparrowright \epsilon\rangle \mathit{tt})$$

$$With \begin{cases} b = (\mathbf{2}\text{-}\boldsymbol{A} \rhd X_v : ((\boldsymbol{A}, \{\{x_v\}_{x_r}\}_{x_b}), \{\{\{x_v\}_{x_r}\}_{x_b}\}_{\boldsymbol{ska}})) \\ b' = (\mathbf{2}\text{-}\boldsymbol{A} \rhd X_v : ((\boldsymbol{A}, \{\{x_v\}_{x'_r}\}_{x'_b}), \{\{\{x_v\}_{x'_r}\}_{x'_b}\}_{\boldsymbol{ska}})) \end{cases}$$

Actions b and b' are sending operations in which the authority A sends a token to the same voter. Thus, the formula ϕ expresses the fact that each legitimate voter is authorized to obtain only one token. Note that, ϕ is a subpart of the formula $\mathcal{I}_P$ specified in the previous section (subformula of the eligibility property), and presented here in a slightly different way.

To prove that the trace t satisfies the formula ϕ, we have to transform first ϕ according to the standard abbreviations of formulae. Thus, we have:

$$\neg(\langle x.b.y.b'.z \looparrowright \epsilon\rangle \mathit{tt}) \equiv [x.b.y.b'.z \looparrowright \epsilon]\neg\nu X.X$$

Now, proving that the formula ϕ is satisfied by the trace t, amounts to show that the sequent $\emptyset, \epsilon, \emptyset, \emptyset \vdash t \in [x.b.y.b'.z \looparrowright \epsilon]\neg\nu X.X$ leads to a successful leaf. This is proved directly, given that the following condition is satisfied:

$$\theta = (H,\ \epsilon,\ e,\ \sigma \vdash t \in [p_1 \looparrowright p_2]\phi) \text{ and } \{\sigma' | p_1\sigma\sigma' = t\} = \emptyset.$$

Indeed, there is no substitutions σ such that $(x.b.y.b'.z)\sigma = t$. We conclude that the particular trace t satisfies the formula ϕ.

An implementation of the tableau-based proof system would allow to check automatically the satisfaction relation $t \models \phi$ for every trace t of the FOO protocol. If we find that a trace of the FOO protocol does not satisfy a given property then we can conclude that the FOO protocol does not guarantee this property (it is the case for the universal verifiability property).

8 Conclusions and Future Work

In this paper we have shown that the ADM logic constitutes a promising candidate for the specification of electronic voting properties. Indeed, thanks to the ADM logic modalities, we have specified four electronic voting properties (fairness, eligibility, individual and universal verifiability), and applied them to the FOO protocol. Moreover, from our analysis of the FOO protocol, we have formally proved in Section 6.4 that this protocol does not satisfy the universal verifiability property.

As future work, we plan to investigate the specification of anonymity and receipt-freeness properties, and to give a more complete analysis of the FOO protocol. This can be done by implementing the rules of the tableau-bases proof system in order to check automatically the specified properties against specific traces (generated by privileging some specific scenarios from the model associated with the FOO protocol).

References

1. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) ASIACRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
2. Juang, W.S., Lei, C.L.: A secure and practical electronic voting scheme for real world environments. TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems (1997)
3. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
4. Benaloh, J.C.: Verifiable secret-ballot elections. PhD thesis, Yale University (1987)
5. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
6. Adi, K., Debbabi, M., Mejri, M.: A new logic for electronic commerce protocols. Theoretical Computer Science 291(3), 223–283 (2003)
7. Stirling, C.: Modal and temporal logics for processes. In: Proceedings of the VIII Banff Higher order workshop conference on Logics for concurrency: structure versus automata, pp. 149–237. Springer, Heidelberg (1996)
8. Baskar, A., Ramanujam, R., Suresh, S.P.: Knowledge-based modelling of voting protocols. In: Proceedings of TARK 2007, pp. 62–71. ACM, New York (2007)
9. Chothia, T., Orzan, S., Pang, J., Dashti, M.T.: A framework for automatically checking anonymity with μ-CRL. In: TGC, pp. 301–318 (2006)
10. Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: Proceedings of CSFW 2006, pp. 28–42. IEEE Computer Society, Los Alamitos (2006)
11. Eijck, J.V., Orzan, S.: Epistemic verification of anonymity. Electronic Notes in Theoretical Computer Science 168, 159–174 (2007)
12. Jonker, H., Pieters, W.: Receipt-freeness as a special case of anonymity in epistemic logic. In: IAVoSS Workshop On Trustworthy Elections - WOTE 2006 (2006)
13. Kremer, S., Ryan, M.: Analysis of an electronic voting protocol in the applied Pi calculus. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 186–200. Springer, Heidelberg (2005)
14. Mauw, S., Verschuren, J., de Vink, E.P.: Data anonymity in the FOO voting scheme. Electronic Notes in Theoretical Computer Science 168, 5–28 (2007)
15. Bergstra, J.A., Klop, J.W.: Algebra of communicating processes with abstraction. Theoritical Computer Science 37, 77–121 (1985)
16. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. ACM SIGPLAN Notices 36(3), 104–115 (2001)
17. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, Cambridge (1999)
18. Pratt, V.R.: Application of modal logic to programming. Studia Logica 39(2-3), 257–274 (1980)
19. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–90 (1981)
20. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security 6(1-2), 85–128 (1998)
21. Dolev, D., Yao, A.C.C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–207 (1981)

Publicly Verifiable Remote Data Integrity

Ke Zeng

NEC Laboratories, China
zengke@research.nec.com.cn

Abstract. More and more customers are outsourcing data storage to remote archive service providers that are responsible for properly preserving the data. As such, it has become crucial for an archive service to be capable of providing evidence to demonstrate the integrity of data for which it is responsible, from the time it receives the data until the expiration of the archival period. Pairing-based provable data integrity (PDI) scheme is proposed that enables not only the customer but also a third-party verifier to check remote data integrity. This PDI scheme is provably secure and efficient. Compared to the best-known prior art, our experiments under defined conditions show that our PDI scheme works 50 times faster in fingerprinting the data, and the resulting fingerprints are 30 times smaller in size.

Keywords: data outsourcing, integrity, public verifiability, pairing.

1 Introduction

Data storage outsourcing is a current trend on the Internet, in which data is stored with a global archive service instead of using one's own local storage. Internet-based online archive services are now providing their end users, including individual customers and businesses, huge amounts of storage space. For example, Apple's iDisk [1] provides 10 GB of online storage space to each .Mac member. Amazon Simple Storage Service (Amazon S3) [2] goes even further. It provides a web services interface that can be used to store and retrieve an unlimited amount of data, with fees metered in GB-months and data-transfer amounts.

Yet with more and more archive services available to store customers' digital assets such as photos, videos, emails, and file system backups, the security of the services that retain this tremendous amount of data becomes critical. As summarized in the IETF's "Long-Term Archive Service Requirements" (RFC 4810) [3], security requirements for archive services must include non-repudiation of data existence, integrity, and origin. Of these requirements, we will herein focus on data integrity. Various news reports have already revealed that even the most popular service providers may damage customers' data [4], such as emails or photos, which may have great personal value. Indeed, archive services are vulnerable to three classes of data integrity threats, namely latent faults (e.g., caused by a bit error in the storage medium), correlated faults (e.g., caused by a lack of geographic location diversity), and recovery faults (e.g., caused by improperly debugged procedures) [4]. Thus, archive service customers, in making the decision to outsource particular data, must be able to evaluate the risk of losing that data.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 419–434, 2008.

One solution is to empower the customers with tools that can help them periodically conduct integrity checks of their data. Another solution, as both Shah et al. [4] and Ateniese et al. [5] proposed, is to introduce a third-party verifier to whom the customers could delegate the periodic task of checking data integrity. More interestingly, the third-party verifier, who has expertise and capabilities that the customers do not, can act as an external auditor of the archive service providers. He can periodically check the integrity of all the data stored with the archive service providers and then release an audit report. Based on the audit report, customers can evaluate the risks associated with any particular archive service provider before they decide to rely on its service. The audit report may also be beneficial to the archive service provider. A positive audit report from a third party may assist the archive service provider in obtaining a favorable insurance rate [4].

It is therefore a must to develop tools for customers, the third-party verifier, and the archive service provider, such that the archive service provider can prove the integrity of data for which it is responsible, from the time it receives the data until the expiration of the archival period [3].

1.1 Related Work

There is a simple solution to tackle the data integrity issue. Initially, the data source divides the data into multiple fragments and for each fragment pre-computes a message authentication code (MAC). Whenever a verifier, be it the data source or a third party, needs to check data integrity, he retrieves from the archive service provider a number of randomly selected fragments and re-computes the MAC of each fragment for comparison. This simple solution has a drawback that its communication complexity is linear with respect to the queried data size. Moreover, in the case of a third-party verifier, sending user data to the verifier is prohibitive because it violates the data source's privacy. To avoid retrieving data from the archive service provider, one may improve this simple solution by choosing multiple secret keys and pre-computing multiple keyed-hash MACs for the data. Thus the verifier can each time reveal a secret key to the archive service provider and ask for a fresh keyed-hash MAC for comparison. However, in this method, the number of times a particular data item can be verified is limited by the number of secret keys that must be fixed a priori. When all possible secret keys are exhausted, it is then necessary to retrieve data from the archive service provider in order to compute new MACs.

Golle et al. [6] proposed a scheme to verify data storage commitment, a concept that is weaker than integrity. The drawback to their proposal is that the verifier's public key is about twice as large as the data file.

Juels et al. [7] proposed to verify data retrievability, a concept that is similar to integrity, by first encrypting the data file then embedding disguised blocks (so-called sentinels) in the ciphertext. One drawback to their proposal is that it is not data format independent, i.e., only encrypted data files can be handled. Hence their proposal is not applicable to a general archive system. Another drawback is that it allows only a limited number of challenges on the data files, which is determined by the number of sentinels embedded at the preprocessing phase.

Schwarz et al. [8] proposed to verify data integrity using an algebraic signature. The drawback to their proposal is that the communication complexity is linear with respect to the queried data size. In addition, the security of their proposal is not proven and remains in question.

Deswarte et al. [9] and Filho et al. [10] proposed to verify data integrity using an RSA-based hash function. Their proposals have the drawback that the archive service provider has to exponentiate the entire data file. As a reference, given a 2048-bit RSA modulus, MIRACL library v5.3.1 [11] reports that one modular exponentiation with a 2048-bit exponent takes 21.8 milliseconds on an Intel Core2 Quad 2.66 GHz processor[1]. Thus it would take 5715 seconds to generate one integrity proof for a merely 64-MB data file. In addition, the security of their proposals remains in question. There is no clear security reduction to the RSA problem or any well-known variant [7].

Further to [9] and [10], Sebe et al. [12] proposed to verify data integrity by first fragmenting the data, fingerprinting each fragment, and then using an RSA-based hash function on the fragments. Their proposal does not require the exponentiation of the entire file. However, the verifier has to have a local copy of the fingerprints, whose size is linear to the number of fragments. In addition, the verifier must not be a third party. Otherwise the secret information of the data source is divulged.

Yamamoto et al. [13] proposed to verify data integrity through batch verification of homomorphic hash functions on randomly selected fragments of data. The drawback to their proposal is that the verifier has to have a local copy of the hash values, whose size is linear to the number of fragments.

Shah et al. [4] proposed allowing a third-party verifier by first encrypting the data then sending a keyed hash of the encrypted data to the verifier. One drawback to their proposal is that the number of times a particular data item can be verified is limited by the number of secret keys that must be fixed beforehand. Another drawback is that their proposal is not data format independent.

Ateniese et al. [5] proposed an S-PDP scheme (where “S” stands for “sampling”), and an S-PDP-PV scheme (where “PV” stands for “public verifiability”) that allows a third-party verifier. Both schemes verify data integrity by first fragmenting the data, then fingerprinting each fragment. The data integrity proof is computed, exploiting the homomorphism of the fingerprints (so-called homomorphic verifiable tags). Both schemes are provably secure and data format independent. They do not require a local copy of the data or the fingerprints and do not confine in advance the maximum number of queries. This sampling strategy introduced by Ateniese et al. is particularly beneficial in the third-party auditing scenario, where the archive service provider needs to prove the integrity of huge volumes of data, e.g., 1 TB of data, to an auditor. Sampling releases the archive service provider from having to access its entire storage, thus largely reducing the required disk I/O overhead. In addition, Ateniese et al. proposed to simplify the S-PDP scheme which yields E-PDP scheme. The E-PDP scheme with weaker security guarantees, i.e., only guarantees possession of the sum of the file blocks, is alleged more efficient than the S-PDP scheme [5].

[1] Throughout this paper, MIRACL reference speeds are always measured using only 1 of the processor’s 4 CPU cores in Windows Vista 32-bit edition.

Unfortunately, both the S-PDP-PV scheme and the E-PDP scheme are unsatisfactory. The S-PDP-PV scheme has the drawbacks that it generates fingerprints that are too large in size and the time it takes to generate the fingerprints is too long as well. As a concrete example, given a 2048-bit RSA modulus, the RSA public key e is chosen to be a 6168 bits prime[2]. The S-PDP-PV scheme mandates that each fragment be less than $e/2$, otherwise the S-PDP-PV scheme is not provably secure. So a 64-MB data file would be divided into at least 87,056 fragments, each of which has a 2048-bit-long fingerprint. Thus, the combined size of all the fingerprints is about 21.2 MB. Further, given a 2048-bit RSA modulus, MIRACL library v5.3.1 reports that one RSA decryption takes 7 milliseconds. Since generating one fingerprint is computationally expensive than doing two RSA decryptions, it would take the data source at least 1218 seconds to fingerprint all the fragments of the 64-MB file.

In terms of the E-PDP scheme, Ateniese et al. [5] argued that its weaker security guarantee is of no practical issue. However, in Section 4.2 we will prove that the E-PDP scheme has no efficiency gain in practice.

To summarize, it seems rational to aim for a third-party-verifier-friendly data integrity proof protocol that satisfies the following four requirements:

1. The verifier does not need a copy of the data or fingerprints.
2. It is provably secure.
3. It is data format independent.
4. The number of allowable queries is not limited in advance.

In addition, an efficient data integrity proof protocol should take the following five factors into consideration:

1. The size of the public key
2. The computation cost for fingerprinting the data
3. The size of the fingerprints
4. The computation cost due to each protocol instance
5. The amount of communication required by each protocol instance

From this viewpoint, the S-PDP-PV scheme is the only solution we are aware of that achieves public verifiability, but it is inefficient due to its excessively high storage and computation consumption.

Our Contributions

We propose a pairing-based provable data integrity (PDI) scheme that is third-party-verifier-friendly and efficient, i.e., it satisfies all the nine requirements above.

Moreover, we implement the PDI scheme to demonstrate its efficiency. In particular, we show experimentally that it takes the PDI scheme only 22.4 seconds to fingerprint one 64-MB file and the fingerprints collectively consume 713 KB. Compared to the S-PDP-PV scheme (under defined conditions above), the PDI scheme is more than 50 times faster in generating the fingerprints and the fingerprints themselves are 30 times smaller in size.

[2] Here we choose a 6168 bits prime e for the S-PDP-PV scheme because it would result in data integrity proof that consumes 771 bytes. We will explain and justify this choice in Section 5.

2 Notations and Number-Theoretic Preliminaries

We first define some notations and review a few number-theoretic preliminaries.

2.1 Notations

If $\mathcal{S}$ is a finite set, $x \in_R \mathcal{S}$ denotes that x is chosen from $\mathcal{S}$ uniformly at random. For two algorithms $A(\cdot)$ and $B(\cdot)$, $(x; y) \leftarrow (A \,||\, B)(\varsigma)$ denotes the joint execution of $A(\cdot)$ and $B(\cdot)$ on the same input string ς and the same random tape, and $A(\cdot)$'s output is assigned to x and $B(\cdot)$'s to y. Let $\Omega(\cdot)$ be an arbitrary Boolean predicate, i.e., a function that upon input of some string ς outputs either TRUE or FALSE. By $\varsigma \leftarrow A(x) : \Omega(\varsigma)$ we denote that $\Omega(\varsigma)$ is TRUE after ς was obtained by running algorithm $A(\cdot)$ on input x.

A function $adv(\kappa)$ is said to be negligible (in κ) if for every positive polynomial $p(\cdot)$ and sufficiently large κ, $adv(\kappa) < 1 / p(\kappa)$.

2.2 Number-Theoretic Preliminaries

Throughout this paper, we use the traditional multiplicative group notation, instead of the additive notation often used in elliptic curve settings.

Let $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ be two finite cyclic groups with additional group $\mathcal{G} = \langle \mathcal{g} \rangle$ such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathcal{G}| = p$ where p is a large prime. Bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathcal{G}$ is a function, such that it is: bilinear – for all $h_1 \in \mathbb{G}_1$, $h_2 \in \mathbb{G}_2$, and for all $a, b \in \mathbb{Z}_p$, $e(h_1^{\ a}, h_2^{\ b}) = e(h_1, h_2)^{ab}$; non-degenerate – $\exists h_1 \in \mathbb{G}_1$, $\exists h_2 \in \mathbb{G}_2$ such that $e(h_1, h_2) \neq \mathcal{I}$ where $\mathcal{I}$ is the identity element of $\mathcal{G}$; and computable – there exists an efficient algorithm for computing e.

We suppose there is a setup algorithm $Setup(\cdot)$ that, upon input of security parameter 1^κ, outputs the above settings of the bilinear map and writes this as $(p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e) \leftarrow Setup(1^\kappa)$. We omit $\mathcal{g}$ from the expression as it is easy to see that $\mathcal{g} = e(g_1, g_2)$.

q-SDH Assumption. For all probabilistic polynomial time (p.p.t.) adversaries $\mathcal{A}$, $adv(\kappa)$ defined as follows is a negligible function:

$$\Pr\left[\begin{array}{c}(p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e) \leftarrow Setup(1^\kappa); a \in_R \mathbb{Z}_p; \\ (x, y) \leftarrow \mathcal{A}(g_2^{\ a}, g_2^{\ a^2}, \cdots, g_2^{\ a^q}) : x \in \mathbb{Z}_p \wedge y = g_1^{\ 1/(a+x)}\end{array}\right] = adv(\kappa)$$

The q-SDH assumption has been shown to hold in generic bilinear groups by Boneh et al. [14].

KEA1 Assumption. For any p.p.t. adversaries $\mathcal{A}$, there exists a p.p.t. extractor $\mathcal{E}$, such that $adv(\kappa)$ defined as follows is a negligible function:

$$(p,\mathbb{G}_1,\mathbb{G}_2,\mathcal{G},g_1,g_2,e) \leftarrow Setup(1^\kappa); x \in_R \mathbb{Z}_p; h \in_R \mathbb{G}_1;$$
$$\Pr\left[(Y,C;\alpha) \leftarrow (\mathcal{A} \,||\, \mathcal{E})(h^x) : Y \in \mathbb{G}_1 \wedge Y = C^x \wedge C \neq h^\alpha\right] = adv(\kappa)$$

Informally, the KEA1 assumption says that the only way any adversary can output $Y = C^x$ from h^x is to pick some $\alpha \in \mathbb{Z}_p$, let $C = h^\alpha$ and let $Y = (h^x)^\alpha$.

q-KEA Assumption. For any p.p.t. adversaries $\mathcal{A}$, there exists a p.p.t. extractor $\mathcal{E}$, such that $adv(\kappa)$ defined as follows is a negligible function:

$$(p,\mathbb{G}_1,\mathbb{G}_2,\mathcal{G},g_1,g_2,e) \leftarrow Setup(1^\kappa); x \in_R \mathbb{Z}_p; \left\{h_j\right\} \in_R \mathbb{G}_1^{\,q};$$
$$\Pr\left[(Y,C;\left\{\alpha_j\right\}) \leftarrow (\mathcal{A} \,||\, \mathcal{E})(\left\{h_j^{\,x}\right\}) : Y \in \mathbb{G}_1 \wedge Y = C^x \wedge C \neq \prod_j h_j^{\,\alpha_j})\right] = adv(\kappa)$$

In the example case of $q = 2$, the 2-KEA assumption (referred to as KEA3 assumption in [15] and XKEA assumption in [16]), says that the only way any adversary can output $Y = C^x$ from $h_1^{\,x}$ and $h_2^{\,x}$ is to pick some $(\alpha_1,\alpha_2) \in \mathbb{Z}_p^{\,2}$, let $C = h_1^{\,\alpha_1} h_2^{\,\alpha_2}$ and let $Y = (h_1^{\,x})^{\alpha_1} (h_2^{\,x})^{\alpha_2}$.

It is proved in [15] that the KEA3 assumption is a natural extension of KEA1. Following that, the q-KEA assumption is trivially provable as a natural extension of the KEA1 assumption as well.

The KEA1 assumption has been shown, by Abe et al. [16] and also by Dent [17] independently, to hold in generic (bilinear) groups. In [16], Abe et al. further proved Lemma 1 as shown below.

Lemma 1. Under the KEA1 assumption, for any p.p.t. adversaries $\mathcal{A}$, there exists a p.p.t. extractor $\mathcal{E}$, such that $adv(\kappa)$ defined as follows is a negligible function:

$$(p,\mathbb{G}_1,\mathbb{G}_2,\mathcal{G},g_1,g_2,e) \leftarrow Setup(1^\kappa); x \in_R \mathbb{Z}_p; \left\{h_1,h_2\right\} \in_R \mathbb{G}_1^{\,2};$$
$$\Pr\left[\begin{array}{l}(\left\{Y_i\right\},\left\{C_i\right\};\left\{\alpha_{i1},\alpha_{i2}\right\}) \leftarrow (\mathcal{A} \,||\, \mathcal{E})(h_1^{\,x},h_2^{\,x}) : \\ \qquad \exists i(Y_i \in \mathbb{G}_1 \wedge Y_i = C_i^{\,x} \wedge C_i \neq h_1^{\,\alpha_{i1}} h_2^{\,\alpha_{i2}}), i = 1,2,\ldots,n\end{array}\right] = adv(\kappa)$$

Lemma 2, below, is a natural extension of Lemma 1. The proof technique for Lemma 1 applies to Lemma 2 as well.

Lemma 2. Under the KEA1 assumption, for any p.p.t. adversaries $\mathcal{A}$, there exists a p.p.t. extractor $\mathcal{E}$, such that $adv(\kappa)$ defined as follows is a negligible function:

$$(p,\mathbb{G}_1,\mathbb{G}_2,\mathcal{G},g_1,g_2,e) \leftarrow Setup(1^\kappa); x \in_R \mathbb{Z}_p; \left\{h_j\right\} \in_R \mathbb{G}_1^{\,q};$$
$$\Pr\left[\begin{array}{l}(\left\{Y_i\right\},\left\{C_i\right\};\left\{\alpha_{ij}\right\}) \leftarrow (\mathcal{A} \,||\, \mathcal{E})(\left\{h_j^{\,x}\right\}) : \\ \qquad \exists i(Y_i \in \mathbb{G}_1 \wedge Y_i = C_i^{\,x} \wedge C_i \neq \prod_j h_j^{\,\alpha_{ij}}), i = 1,2,\ldots,n\end{array}\right] = adv(\kappa)$$

3 Provable Data Integrity Scheme

We start with the definition and description of our provable data integrity (PDI) scheme, followed by its security analysis, discussion, and performance evaluation.

3.1 Definition

A PDI scheme is a collection of four algorithms, $KeyGen(\cdot)$, $Fingerprint(\cdot)$, $GDIP(\cdot)$, and $VDIP(\cdot)$.

$(pk, sk) \leftarrow KeyGen(1^{\kappa})$. This probabilistic algorithm takes as input security parameter 1^{κ}, and returns public key pk and private key sk.

$(z_k, T_i^k) \leftarrow Fingerprint(pk, sk, \mathcal{F}_k)$. This algorithm takes as input public key pk, private key sk, and a file $\mathcal{F}_k$. Let FN_k denote $\mathcal{F}_k$'s file reference, which is, for example, the file name of $\mathcal{F}_k$ plus a unique serial number. The file $\mathcal{F}_k$ is an ordered collection of super-blocks M_i^k while each super-block M_i^k is an ordered collection of file blocks m_j^{ik}. This algorithm returns a file key z_k for $\mathcal{F}_k$ and the fingerprint T_i^k for M_i^k. There is a one-to-one mapping between z_k and FN_k. The length of the file block and the number of file blocks that one super-block can aggregate are determined by certain parameters of pk.

$\mathrm{V} \leftarrow GDIP(pk, FN, \mathcal{F}, \mathrm{T}, chal)$. This algorithm takes as input public key pk, a file $\mathcal{F}$ that is an ordered collection of super-blocks M_i, the file reference FN of $\mathcal{F}$, an ordered collection of fingerprints $\mathrm{T} = \{T_i\}$ for $\{M_i\}$, and a challenge $chal$. It returns a data integrity proof V.

$\{\mathrm{TRUE}, \mathrm{FALSE}\} \leftarrow VDIP(pk, FN, z, chal, \mathrm{V})$. This algorithm takes as input public key pk, a file reference FN, a file key z, a challenge $chal$, and the data integrity proof V. It returns TRUE if the integrity of the file $\mathcal{F}$ determined by FN is verified as correct, or FALSE otherwise.

Based on the PDI scheme, a PDI system could be easily constructed in three phases, **Setup**, **Store**, and **Challenge**.

Setup: A client's public key and private key are initialized by invoking $KeyGen(\cdot)$. The client publishes his public key.

Store: A client in possession of a file runs $Fingerprint(\cdot)$ to fingerprint the file and stores the file and its fingerprints with the server. The client deletes the file and the fingerprints from his local storage.

Challenge: Any one, a third party or the client himself, can verify integrity of the client's file by invoking $VDIP(\cdot)$, which requires to challenge the server and the server executing $GDIP(\cdot)$ to respond. It is notable that this phase can be executed an unlimited number of times.

3.2 The PDI Scheme

Now we start to depict our PDI scheme. For simplicity, we regard $(p, \mathbb{G}_1, \mathbb{G}_2, \mathcal{G}, g_1, g_2, e) \leftarrow Setup(1^\kappa)$ and pseudo random functions $prf_1 : \{0,1\}^* \to \mathbb{Z}_p$, $prf_2 : \{0,1\}^* \to \mathbb{G}_1$, $prf_3(\phi) : \{0,1\}^* \to \mathbb{Z}_{2^\phi}$ as system parameters.

***KeyGen(·)*:** Select $sk = (x,s,t) \in_R \mathbb{Z}_p{}^3$. Compute $Y = g_2{}^x$, $Y_s = Y^s$, $g_{2s} = g_2{}^s$, $h_j = g_1{}^{prf_1(j,t)}$, and $S_j = h_j{}^s$, $j = 1,2,\ldots,n_B$. Choose a positive integer $len < \log p$ that determines the length in bits of each file block and another positive integer n_B which determines the number of file blocks that one super-block contains.

Finally, output $sk = (x,s,t)$ and $pk = \left(Y, Y_s, g_{2s}, \left\{h_j, S_j\right\}, len, n_B\right)$.

Given pk, a file $\mathcal{F}$ with file reference FN can be divided into $N = \lceil Flen / len \rceil$ file blocks m_i, each of which is len bits long, where $Flen$ is the length of $\mathcal{F}$ in bits. And every n_B consecutive file blocks constitute one super-block. In other words, the file $\mathcal{F}$ is logically divided into N file blocks and organized into $n_{SB} = \lceil N / n_B \rceil$ super-blocks. Notice that the file $\mathcal{F}$ will be logically padded with zero in the case that $Flen < N \cdot len$ or $Flen < n_{SB} \cdot (n_B \cdot len)$.

***Fingerprint(·)*:** Upon input of $sk = (x,s,t)$, $pk = \left(Y, Y_s, g_{2s}, \left\{h_j, S_j\right\}, len, n_B\right)$, and a file $\mathcal{F}$ with file reference FN, select $z \in_R \mathbb{Z}_p$ as $\mathcal{F}$'s file key and execute the following for each super-block M_i of $\mathcal{F}$, $i = 1,2,\ldots,n_{SB}$:

a) Compute a locator $W_i = prf_2(i,z,FN) \in \mathbb{G}_1$.

b) Compute $R_i = \sum_{j=1}^{n_B} prf_1(j,t) \cdot m_{(i-1)\cdot n_B + j} \in \mathbb{Z}_p$ and $T_i = (W_i \cdot g_1{}^{R_i})^{\frac{1}{x+z}} \in \mathbb{G}_1$.

Finally, output the file key z and the fingerprints $\mathrm{T} = \left\{T_i\right\} \in \mathbb{G}_1{}^{n_{SB}}$.

Notice that $T_i = (W_i \cdot g_1{}^{R_i})^{\frac{1}{x+z}} = (W_i \cdot \prod_{j=1}^{n_B} h_j{}^{m_{(i-1)\cdot n_B + j}})^{\frac{1}{x+z}}$ for all $i = 1,2,\ldots,n_{SB}$.

***GDIP(·)*:** Upon input of $pk = \left(Y, Y_s, g_{2s}, \left\{h_j, S_j\right\}, len, n_B\right)$; a file $\mathcal{F}$ with file reference FN, file key z, and fingerprints $\mathrm{T} = \left\{T_i\right\}$; $chal = (l, \psi, \gamma_1, \gamma_2)$, where $1 \le \psi \le l$ and $(\gamma_1, \gamma_2) \in_R \mathbb{Z}_p{}^2$; execute the following:

a) Compute $\phi = \left\lceil l / \psi \right\rceil + 1$.

b) For each $k = 1, 2, \ldots, \psi$ execute the following atomic proof procedure once (i.e., repeat the atomic proof procedure independently ψ times).

   1) Consider there to be $\Phi = 2^{\phi}$ buckets in logic. For each bucket, initialize a packed fingerprint $\vec{T}_v = \mathcal{O}$, and packed file blocks $e_{vj} = 0$, where $\mathcal{O}$ is the identity of $\mathbb{G}_1$, $v = 0, 1, \ldots, \Phi - 1$, $j = 1, 2, \ldots, n_B$.

   2) For each super-block M_i and its corresponding fingerprint T_i, randomly assign them into one bucket and add them to the bucket's packed file blocks and packed fingerprint, respectively. Specifically, for each $i = 1, 2, \ldots, n_{SB}$, conduct the following:

      i. Compute $\sigma = prf_3(i, k, \gamma_1) \in \mathbb{Z}_{2^{\phi}}$.

      ii. Compute $\vec{T}_\sigma \ {*}{=}\ T_i$ (i.e., compute $\vec{T}_\sigma \cdot T_i$ and store back to $\vec{T}_\sigma$).

      iii. For each $j = 1, 2, \ldots, n_B$, compute $e_{\sigma j} \ {+}{=}\ m_{(i-1)\cdot n_B + j} \bmod p$.

   3) Initiate a transformed fingerprint $\mathrm{T}_k = \mathcal{O} \in \mathbb{G}_1$ and transformed file blocks $E_j = 0$, $j = 1, 2, \ldots, n_B$.

   4) Assign each bucket a random number to randomize its packed fingerprint and packed file block, then add them to the transformed fingerprint and the transformed file blocks, respectively. Specifically, for each $v = 0, 1, \ldots, \Phi - 1$, conduct the following:

      i. Compute $a_v = prf_3(v, k, \gamma_2) \in \mathbb{Z}_{2^{\phi}}$.

      ii. Compute $\mathrm{T}_k \ {*}{=}\ \vec{T}_v^{\,a_v}$.

      iii. For each $j = 1, 2, \ldots, n_B$, compute $E_j \ {+}{=}\ a_v \cdot e_{vj} \bmod p$.

   5) Compute $\mathrm{H}_k = \prod_{j=1}^{n_B} S_j^{E_j} \in \mathbb{G}_1$ as the knowledge proof of the transformed file block T_k.

Finally, output the data integrity proof $(\mathrm{T}_k, \mathrm{H}_k)$, $k = 1, 2, \ldots, \psi$ [3].

VDIP(·): Upon input of $pk = \left(Y, Y_s, g_{2s}, \left\{h_j, S_j\right\}, len, n_B\right)$; a file reference FN; a file key z; $chal = (l, \psi, \gamma_1, \gamma_2)$, where $1 \le \psi \le l$ and $(\gamma_1, \gamma_2) \in_R \mathbb{Z}_p^{\,2}$; and a data integrity proof $(\mathrm{T}_k, \mathrm{H}_k) \in \mathbb{G}_1^{\,2}$, $k = 1, 2, \ldots, \psi$; execute the following:

[3] The size of the data integrity proof is proportional to the number of atomic proof procedures. However this is of little efficiency concern in practice. See Section 5 for detailed discussion on this matter.

a) For each $k = 1, 2, \ldots, \psi$, execute the following atomic verification procedure once (i.e., repeat the atomic verification procedure independently ψ times).

1) Initialize a transformed locator $\mathrm{W}_k = \mathcal{O}$.

2) Consider there to be $\Phi = 2^{\phi}$ buckets in logic. For each bucket, initialize a packed locator $\vec{W}_v = \mathcal{O}$, $v = 0, 1, \ldots, \Phi - 1$.

3) Re-compute the locators, randomly assign them into the buckets, and add them to the bucket's packed locators. Specifically, compute $\sigma = prf_3(i, k, \gamma_1) \in \mathbb{Z}_{2^{\phi}}$ and $\vec{W}_{\sigma}$ $*=$ $prf_2(i, z, FN)$ for each $i = 1, 2, \ldots, n_{SB}$.

4) Assign the buckets random numbers to randomize their packed locators, then add them to the transformed locator. Specifically, compute $a_v = prf_3(v, k, \gamma_2) \in \mathbb{Z}_{2^{\phi}}$ and W_k $*=$ $\vec{W}_v^{\,-a_v}$ for each $v = 0, 1, \ldots, \Phi - 1$.

5) If $e(\mathrm{H}_k, g_2) = e(\mathrm{T}_k, Y_s \cdot g_{2s}^{\ z}) \cdot e(\mathrm{W}_k, g_{2s})$, the output is TRUE, otherwise the output is FALSE.

$VDIP(\cdot)$ outputs TRUE if and only if all the atomic verification procedures output TRUE.

3.3 Security of the PDI Scheme

Informally, the security of this PDI scheme is equivalent to the nonexistence of an adversary that is capable, within the confines of a certain game, of forging the data integrity proof on the condition that at least one file block is not present. We define the security of the PDI scheme as an adaptive chosen-file-block game. In this model, the adversary $\mathcal{A}$ is given a single public key. His goal is to output a data integrity proof. We give the adversary the power to choose all the file blocks as well as the super-blocks. The adversary is also given oracle access to fingerprint issuance on the super-blocks.

Remote Data Integrity Game

Setup. The challenger runs $(pk, sk) \leftarrow KeyGen(1^{\kappa})$, sends pk to the adversary, and keeps sk secret.

Queries. The adversary $\mathcal{A}$ makes fingerprint queries adaptively; it selects file blocks m_j^{ik} that form super-block M_i^k, assigns M_i^k to a file $\mathcal{F}_k$ that is referenced by FN_k, and sends FN_k and M_i^k to the challenger. The challenger computes $(z_k, T_i^k) \leftarrow Fingerprint(pk, sk, FN_k, M_i^k)$, stores z_k, and sends T_i^k back to $\mathcal{A}$. The file key z_k is not revealed to the adversary at this phase. The challenger needs to ensure one-to-one mapping between z_k and FN_k. One restriction on $\mathcal{A}$ is that it must not query different super-blocks with the same indexes k and i. Another restriction on $\mathcal{A}$ is that the length of the file block and the number of file blocks that one super-block has aggregated must comply with those being determined by the parameters of pk .

Challenge. The adversary selects a file reference FN that determines a file $\mathcal{F}$. The file $\mathcal{F}$ is an ordered collection of super-blocks M_i, each of which has a fingerprint T_i . And each super-block M_i is an ordered collection of file blocks m_j^i . The challenger first outputs z that is the file key of FN , then generates a challenge $chal$ and asks the adversary for a data integrity proof.

Restricted Queriesx. The adversary $\mathcal{A}$ is allowed to continue querying fingerprints as before, except for when adding a new super-block to $\mathcal{F}$ and querying the fingerprint for the new super-block. In other words, the adversary is not allowed to expand the content of $\mathcal{F}$.

Output. The adversary $\mathcal{A}$ outputs a data integrity proof V .

The adversary wins the game if $\mathrm{TRUE} \leftarrow VDIP(pk, FN, z, chal, \mathrm{V})$.

Definition (Security of PDI Scheme). The PDI scheme is secure if for any p.p.t. adversary $\mathcal{A}$ the probability that $\mathcal{A}$ wins the Remote Data Integrity Game on a set of file blocks is negligibly close to the probability that the challenger can extract the file blocks by means of a knowledge extractor $\mathcal{E}$.

Theorem 1. The PDI scheme that achieves public verifiability is secure under the q-SDH assumption and the KEA1 assumption in the random oracle model.

4 Discussions

4.1 Sampling for the PDI Scheme

We can easily add the sampling strategy to the PDI scheme, yielding an S-PDI scheme. The S-PDI scheme in addition requires a $prf_4(n_{SB}) : \{0,1\}^* \to \mathbb{Z}_{n_{SB}}$ and the challenge will contain a third key $\gamma_3 \in_R \mathbb{Z}_p$ and a positive integer $\Lambda < n_{SB}$. By computing $\mathrm{i}_1 = prf_4(1, \gamma_3)$, $\mathrm{i}_2 = prf_4(2, \gamma_3)$, up to $\mathrm{i}_\Lambda = prf_4(\Lambda, \gamma_3)$, the indexes of Λ randomly selected super-blocks are determined. Each atomic proof/verification procedure will then only deal with the super-blocks being selected. This entails no more than changing all $i = 1, 2, \ldots, n_{SB}$ in $GDIP(\cdot)$ and $VDIP(\cdot)$ to $i = \mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_\Lambda$.

Theorem 2. The S-PDI scheme that achieves public verifiability is secure under the q-SDH assumption and the KEA1 assumption in the random oracle model.

4.2 Knowledge Error and Its Practical Implication

The standard notion of proofs of knowledge as per [19], specifically, Definition 3 of [19], contains a knowledge error that is the probability that the verifier might accept even if the prover did not in fact “know” the witness. When constructing a proof of knowledge system, it is the knowledge error that determines the number of necessary repetitions in practice. By repetition, the knowledge error could be reduced to an arbitrarily small amount. For example, if the knowledge error is 1/2, Γ times of sequential iterations may reduce the knowledge error to $2^{-\Gamma}$ [20].

Since repetitions linearly increase the computation complexity and communication complexity of a proof of knowledge system, quantifying the knowledge error is a must before analyzing practical efficiency of a proof of knowledge scheme. However, this part was missing in [5] for the E-PDP, S-PDP, and S-PDP-PV schemes.

Claim 1. The E-PDP scheme of [5] attains knowledge error no smaller than 1/2.

Proof (Sketch). The Atomic Random Subset Test method disclosed in [18] could be used to construct a specific attack on the E-PDP scheme. Although both guarantee that congruence $\sum_{i=1}^{c} m_i' = M^* = M = \sum_{i=1}^{c} m_i$ hold, the E-PDP scheme allows a stronger adversary than the Atomic Random Subset Test method does. This is because the verifier by the Atomic Random Subset Test method receives all the file blocks m_i' , whereas, the verifier by the E-PDP scheme receives only M^* . From this viewpoint, what the E-PDP scheme does is no more than first choosing c file blocks as the full set then applying subset test on the full set itself. As per Lemma 3.1[4] of [18], this allows for knowledge error 1/2. □

Recall that in [5] it was shown that a challenge on $c = 460$ randomly selected file blocks can reach a detection probability of 99.02% in the case of failure of 1% of the file blocks. However, calculating this probability has a precondition, i.e., all the c file blocks must be correctly possessed by the archive service with overwhelming probability. If with probability 1/2 that at least one of the c file blocks is incorrect, the E-PDP scheme has to be iterated in practice. For instance, $\Gamma = 15$ reiterations reduce the knowledge error to 2^{-15} , i.e., the probability that at least one of the c file blocks is incorrect is 2^{-15} . Thus the overall detection probability can reach $(1-2^{-15}) * 99.02\% > 99\%$. As the consequence, both the computation complexity and the communication complexity of the E-PDP scheme would be Γ times larger in practice.

Claim 2. The S-PDP scheme (also the S-PDP-PV scheme) of [5] attains knowledge error no smaller than 2^{-l} for each file block, where l determines the bit length of the coefficients a_i 's.

Proof (Sketch). Very briefly, the Small Exponent Test method disclosed in [18] could be used to construct a specific attack on the S-PDP scheme. □

[4] Lemma 3.1 of [18] is originally proved over a finite cyclic group G of prime order p. It's not hard to generalize the proof for Lemma 3.1 to the case of a finite cyclic group of composite order p'q' regardless of whether p'q' is available to the adversary or not, e.g., G=QR(N). Theorem 3.3 of [BRG 98] can be generalized to work on QR(N) as well.

Claim 3. The PDI scheme attains knowledge error no smaller than 2^{-l} for each file block.

Proof (Sketch). Very briefly, the atomic proof procedure and the atomic verification procedure of the PDI scheme are in principle reusing the Bucket Test method, as per the findings of [18]. □

Claim 4. The S-PDP scheme (also the S-PDP-PV scheme) of [5] attains knowledge error no larger than $(c-1)\cdot 2^{-l}$ for each file block, where l determines the bit length of the coefficients a_i 's and $c \geq 2$ is the number of the file blocks being chosen.

Proof (Sketch). Our proof reuses the proof technique that Damgard utilized to prove his Theorem 1 in [21]. □

Claim 5. The S-PDI scheme attains knowledge error no larger than $\frac{c}{\psi}\cdot 2^{-l}$ for each file block, where ψ is the number of repetitions required for the atomic proof/verification procedure and $c \geq \psi$ is the number of the super-blocks being chosen.

5 Efficiency of the PDI Scheme

In order to demonstrate its efficiency, we implement the PDI scheme using MIRACL library v5.3.1 in Windows Vista 32-bit edition. All experiments were conducted on a Dell Precision 390 workstation with an Intel Core2 Quad 2.66-GHz processor, 4 GB of ECC RAM, and a 400-GB RAID-0 disk array consisting of 3 SCSI drives.

The 64-MB data file we use for the experiments is AES-CBC encrypted such that our experiment results are not necessarily subject to the entropy of the data file.

We use a pairing-friendly non-supersingular curve, as per Brezing et al. [22]. Its parameters as listed below are provided by MIRACL library source code, wherein the modulus q is a 256-bit prime, the group order p is a 192-bit prime, and the embedding degree is 8. The security depends on the difficulty of a 2048-bit discrete logarithm problem [11].

$q = \mathrm{CC485D26177A1A5FCC9D53BA93DA298FD7F2F23D8FC02A8123BF24F9548A5F15}$

$p = \mathrm{9D0261DD89CF83D5D20198162C22C942EF68622A6DF25621}$

$A = 0,\ B = 2,\ cof = 14D13FF7298021445,\ k = 8$

We choose SHA-256 as $prf_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and UMAC [23] as $prf_3(\phi) : \{0,1\}^* \rightarrow \mathbb{Z}_{2^\phi}$. And we choose SHA-256 to build $prf_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$.

In order to fingerprint the file, we choose $n_B = 256$ and $len = 184$, thus each super-block contains 5888 bytes and the 64-MB file consists of 11,398 super-blocks.

Our experiments show that the size of the public key is 32.5 KB[5], generating fingerprints takes 22.4 seconds[6], and the size of the fingerprints is about 713 KB.

In order to evaluate the efficiency of generating data integrity proof and verifying the proof, we choose $l = 84$ and $\psi = 12$, which means that the atomic proof/verification procedure will be reiterated 12 times to attain security level $1-\frac{11398}{12}\cdot 2^{-84} > 1-2^{-74}$ for every 184 bits of the 64-MB data file. Each atomic proof/verification procedure needs to handle 256 buckets.

Our experiments show that generating the data integrity proof takes 6.395 seconds, verifying the proof takes 6.961 seconds, the size of the challenge is 50 bytes, and the size of the data integrity proof is 771 bytes.

Recall that the S-PDP-PV scheme needs at least 1218 seconds to fingerprint a 64-MB data file and those fingerprints collectively require 21.2 MB. It is therefore clear that our PDI scheme is more than 50 times faster in generating the fingerprints and the fingerprints themselves are 30 times smaller in size. It is notable that in this comparison we choose a 6168 bits prime e for the S-PDP-PV scheme. As per the S-PDP-PV scheme, this choice results in data integrity proof that consumes 771 bytes ([5], p.14). We note that choosing a larger e for the S-PDP-PV scheme could reduce the size of its fingerprints and the time to generate the fingerprints as well. Whereas a larger e at the same time increases the size of its data integrity proof. We thus compare efficiency of the PDI scheme and the S-PDP-PV scheme on condition that they generate the same size of data integrity proof. This comparison is appropriate at least for the case that the amount of communication required by each protocol instance is strictly restricted.

5.1 Speeding Up the PDI Implementation

Noting that fingerprinting can be done on multiple super-blocks simultaneously, and the atomic proof/verification procedures run independently, parallel computation should therefore be beneficial to the PDI scheme. Utilizing OpenMP [25] technology, we make the fingerprinting, proof generation, and proof verification all run in parallel. Making use of all 4 of the CPU cores of the Intel Core2 Quad 2.66 GHz processor, our experiments show that for the 64-MB file, generating fingerprints takes 5.632 seconds, generating the data integrity proof takes 1.997 seconds, and verifying the proof takes 1.769 seconds. Using the 4 CPU cores yields roughly a three- to fourfold speed increase compared to using only 1 CPU core.

Another approach for speeding up the implementation of the PDI scheme is simply reducing the intended security level. For instance, choosing $l = 48$ and $\psi = 8$ will reduce the security level to $1-2^{-37}$ and our experiments show that generating one integrity proof now takes 1.279 seconds, compared to 1.997 seconds for security level $1-2^{-74}$. It is, however, more interesting to exploit the sampling strategy, i.e., make use of the S-PDI scheme.

[5] Except for storing the fingerprints, we always use point compression technology, by which each elliptic curve point is stored as its x coordinate and the LSB of its y coordinate.

[6] All experimental results on time consumptions represent the mean of 10 trials.

Similar to the analysis given by [5], a challenge on 460 super-blocks can reach a detection probability of 99.02% in the case of failure of 1% of the super-blocks. Therefore, choosing $l = 23$, $\psi = 4$, and 460 super-blocks per challenge, the overall detection probability would be $(1 - \frac{460}{4} \cdot 2^{-23}) * 99.02\% > 99\%$. We implement the S-PDI scheme and our experiments show 0.312 seconds in generating one integrity proof and 0.077 seconds in verifying the proof, when all 4 CPU cores are utilized. Note that in this case, the size of the data integrity proof is mere 257 bytes.

6 Conclusions and Future Work

In this paper, we present a provably secure and efficient scheme, which enables not only the data source but also a third-party verifier to check remote data integrity.

There are still some problems yet to be resolved. First, the PDI scheme is not able to fingerprint data file incrementally. Our Remote Data Integrity Game prohibits a data file been modified in whatever manners once the file has been fingerprinted. Second, in the case that the data source is malicious, a third-party verifier who dares to be responsible for the integrity of the data is in danger because the data source can arbitrarily manipulate the data stored with the archive service provider while the fingerprints remain valid. Third, in the case that a third-party verifier colludes with the archive service provider, one can easily surmise that the verifier can fabricate a favorable audit report for a customer's precious file even if the archive service provider has deleted the file completely. Last but not least, it is always interesting to find novel third-party-verifier-friendly schemes that are in the standard model.

Acknowledgements

The author is indebted to Hao Lei, Ye Tian, Li-Ming Wang, and Yu-Liang Zheng for insightful discussions during the development of the ideas herein presented. The author is particularly grateful to the anonymous referees for their useful comments. The author would like to thank Min-Yu Hsueh and Toshikazu Fukushima for their support to this research.

References

1. Apple iDisk, http://www.apple.com/dotmac/idisk.html
2. Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3
3. Wallace, C., Pordesch, U., Brandner, R.: Long-Term Archive Service Requirement, RFC 4810. IETF Network WG (2007)
4. Shah, M.A., Baker, M., Mogul, J.C., Swaminathan, R.: Auditing to Keep Online Storage Services Honest. In: 11th Workshop on Hot Topics in Operating Systems (HotOS-XI), Usenix (2007)
5. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable Data Possession at Untrusted Stores. In: 14th ACM conference on Computer and Communications Security (CCS 2007), pp. 598–609. ACM Press, New York (2007), http://eprint.iacr.org/2007/202/

6. Golle, P., Jarecki, S., Mironov, I.: Cryptographic primitives enforcing communication and storage complexity. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 120–135. Springer, Heidelberg (2003)
7. Juels, A., Kaliski, B.S.: PORs: Proofs of Retrievability for Large Files. Report 2007/243, Cryptology ePrint archive (2007)
8. Schwarz, T.S.J., Miller, E.L.: Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage. In: IEEE International Conference on Distributed Computing Systems (ICDCS 2006), p. 12. IEEE Press, Los Alamitos (2006)
9. Deswarte, Y., Quisquater, J.J., Saidane, A.: Remote Integrity Checking. In: 6th IFIP TC-11 WG 11.5. In: Working Conference on Integrity and Internal Control in Information Systems (IICIS 2003), pp. 1–11. IFIP Press (2003)
10. Filho, D.L.G., Baretto, P.S.L.M.: Demonstrating Data Possession and Uncheatable Data Transfer. Report 2006/150, Cryptology ePrint Archive (2006)
11. MIRACL, Multi-precision Integer and Rational Arithmetic C Library, `http://www. shamus.ie`
12. Sebe, F., Ferrer, J.D., Balleste, A.M., Deswarte, Y., Quisquater, J.J.: Efficient Remote Data Possession Checking in Critical Information Infrastructures. IEEE Transactions on Knowledge and Data Engineering 20(8), 1034–1038 (2007)
13. Yamamoto, G., Oda, S., Aoki, K.: Fast Integrity for Large Data. In: Workshop on Software Performance Enhancement for Encryption and Decryption (SPEED 2007), pp. 21–32. COSIC Press (2007)
14. 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)
15. Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
16. Abe, M., Fehr, S.: Perfect NIZK with Adaptive Soundness. Report 2006/423, Cryptology ePrint Archive (2006)
17. Dent, A.W.: The Hardness of the DHK Problem in the Generic Group Model. Report 2006/156, Cryptology ePrint Archive (2006)
18. Bellare, M., Garay, J.A., Rabin, T.: Fast Batch Verification for Modular Exponentiation and Digital Signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
19. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
20. Damgard, I., Pfitzmann, B.: Sequential Iteration of Interactive Arguments and an Efficient Zero-Knowledge Argument for NP. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 772–783. Springer, Heidelberg (1998)
21. Damgard, I.: On Σ-protocols, `http://www.daimi.au.dk/~ivan/Sigma.pdf`
22. Brezing, F., Weng, A.: Elliptic Curves Suitable for Pairing Based Cryptography. Report 2003/143, Cryptology ePrint Archive (2003)
23. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: Fast and Secure Message Authentication. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 215–233. Springer, Heidelberg (1999)
24. Kaliski, B.: TWIRL and RSA Key Size. RSA Laboratories Technical Notes and Reports, `http://www.rsa.com/rsalabs/node.asp?id=2004`
25. OpenMP, Open Multi-Processing Application Program Interface (API) Specification for Parallel Programming, `http://www.openmp.org`
26. OpenSSL, The Open Source Toolkit for SSL/TLS, `http://www.openssl.org`

Author Index