Juan A. Garay
Arjen K. Lenstra
Masahiro Mambo
René Peralta (Eds.)

LNCS 4779

Information Security

10th International Conference, ISC 2007
Valparaíso, Chile, October 2007
Proceedings

Springer

Lecture Notes in Computer Science 4779

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Juan A. Garay Arjen K. Lenstra
Masahiro Mambo René Peralta (Eds.)

Information Security

10th International Conference, ISC 2007
Valparaíso, Chile, October 9-12, 2007
Proceedings

Springer

Volume Editors

Juan A. Garay
Bell Labs
600 Mountain Ave., Murray Hill, NJ 07974, USA
E-mail: garay@research.bell-labs.com

Arjen K. Lenstra
EPFL IC LACAL
INJ 330, Station 14, CH-1015 Lausanne, Switzerland
E-mail: arjen.lenstra@epfl.ch

Masahiro Mambo
University of Tsukuba
1-1-1 Tennoudai, Tsukuba, Ibaraki, 305-8573, Japan
E-mail: mambo@cs.tsukuba.ac.ip

René Peralta
NIST, Security Division, Information Technology Laboratory
Gaithersburg, MD. 20899, USA
E-mail: rene.peralta@nist.gov

Library of Congress Control Number: 2007936070

CR Subject Classification (1998): E.3, E.4, D.4.6, C.2, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-75495-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-75495-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12170333 06/3180 5 4 3 2 1 0

Preface

The 10th Information Security Conference (ISC 2007) was held in Valparaíso, Chile, October 9–12, 2007. ISC is an annual international conference covering research in theory and applications of information security, aiming to attract high quality papers in all of its technical aspects. ISC was first initiated as a workshop (ISW) in Japan in 1997, ISW 1999 was held in Malaysia and ISW 2000 in Australia. The name was changed to the current one when the conference was held in Spain in 2001 (ISC 2001). The latest conferences were held in Brazil (ISC 2002), the UK (ISC 2003), the USA (ISC 2004), Singapore (ISC 2005), and Greece (ISC 2006). This year the event was sponsored by the Universidad Técnica Federico Santa María (Valparaíso, Chile), the Support Center for Advanced Telecommunications Technology Research, Foundation, SCAT(Tokyo, Japan), Microsoft Corporation, and Yahoo! Research.

Reflecting the conference's broad scope, this year's main Program Committee consisted of a relatively large number (49) of experts. Additionally, given the timely topic of cryptanalysis and design of hash functions and the NIST hash competition, the conference also featured a special Hash Subcommittee, chaired by Arjen Lenstra (EPFL and Bell Labs), as well as a panel on hashing, chaired by Bill Burr (NIST). The conference received 116 submissions, 29 of which were selected by the committee members for presentation at the conference, based on quality, originality and relevance. Each paper was anonymously reviewed by at least three committee members.

Extended abstracts of 28 of the selected papers (a decision was made that only papers whose authors could commit to presenting them at the conference would be published), many revised according to the reviewers' suggestions, appear in these proceedings. An important ISC interest is to encourage and promote student participation. In line with that interest, the ISC 2007 Program Committee had the pleasure of selecting three student-coauthored papers for the Best Student Paper award—one from each region ISC rotates among: Asia, Europe, and the Americas. The papers were, respectively, "Identity-Based Proxy Re-encryption Without Random Oracles," by Cheng-Kang Chu and Wen-Guey Tzeng (National Chiao Tung University, Taiwan), "Detecting System Emulators," by Thomas Raffetseder, Christopher Kruegel, and Engin Kirda (Technical University of Vienna, Austria), and "Impossible-Differential Attacks on Large-Block Rijndael," by Jorge Nakahara Jr. and Ivan Carlos Pavão (Catholic University of Santos, Brazil). The program also included invited lectures by Hugo Krawczyk (IBM's T.J. Watson Research Center, USA), and Brent Waters (SRI International, USA).

First and foremost, I am extremely grateful to the members of the Program Committee and Hash Subcommittee for their investment and effort in the

process—many times difficult and delicate—of paper review and selection, as well as to the large number of external reviewers for their valuable help.

Electronic submissions were made possible by the Web Submission and Review Software developed by Shai Halevi, which was hosted at the Universidad Técnica Federico Santa María. Many thanks to Raul Monge for making that possible—and for his perennial availability when problems arose, to Shai for his support, and to Debbie Cook and Marcos Kiwi for their help in the handling of the submissions.

Beyond the hosting of the submission software, Raúl Monge and his team did a magnificent job managing and taking care of all aspects of the local organization. I am also most grateful to the general chairs, Masahiro Mambo and René Peralta, for all their hard work, assistance and advice on a myriad of issues related to this conference.

Finally, I wish to thank all the authors for submitting their work to ISC 2007, and the authors of the accepted papers for their contribution to the high technical quality of the program. As technology evolves and means of communication and interaction become increasingly more complex and sophisticated, so does the need not only for guaranteeing their soundness and safety when run in adversarial settings, but also for novel techniques that actually make them possible. Without a doubt, the new notions, methods and designs presented in these proceedings constitute an important step in those directions.

August 2007 Juan A. Garay

ISC 2007

The 10th International Security Conference
Valparaíso, Chile, October 9–12, 2007

ISC Steering Committee

Ed Dawson	Queensland University of Technology, Australia
Sokratis K. Katsikas	University of the Aegean, Greece
Javier López	University of Málaga, Spain
Masahiro Mambo	University of Tsukuba, Japan
Eiji Okamoto	University of Tsukuba, Japan
René Peralta	NIST, USA
Rebecca Wright	Rutgers University, USA
Yuliang Zheng	University of North Carolina–Charlotte, USA

General Chairs

Masahiro Mambo	University of Tsukuba, Japan
René Peralta	NIST, USA

Program Chair

Juan A. Garay	Bell Labs, USA

Hash Subcommittee Chair

Arjen Lenstra	EPFL, Switzerland and Bell Labs, USA

Organizing Chair

Raúl Monge	Universidad Técnica Federico Santa María, Chile

Program Committee

Michel Abdalla	ENS, France
Mikhail Atallah	Purdue University, USA
Michael Backes	Saarland University, Germany
Feng Bao	Institute for Infocomm Research, Singapore
Paulo Barreto	University of Sao Paulo, Brazil

John Black	University of Colorado, USA
Debbie Cook	Bell Labs, USA
Claudia Diaz	K.U. Leuven, Belgium
Glenn Durfee	PARC, USA
Nelly Fazio	IBM Research, USA
Matthias Fitzi	ETH Zürich, Switzerland
Stuart Haber	HP Labs, USA
Shai Halevi	IBM Research, USA
Amir Herzberg	Bar-Ilan University, Israel
Alejandro Hevia	University of Chile, Chile
Trent Jaeger	Penn State University, USA
Stasio Jarecki	University of California-Irvine, USA
Angelos Keromytis	Columbia University, USA
Aggelos Kiayias	University of Connecticut, USA
Kwangjo Kim	Information and Comms. University, Korea
Marcos Kiwi	University of Chile, Chile
Steve Kremer	ENS Cachan, France
Dong Hoon Lee	Korea University, Korea
Helger Lipmaa	University College London, UK
Breno de Medeiros	Florida State University, USA
Atsuko Mijayi	JAIST, Japan
Fabian Monrose	Johns Hopkins University, USA
Gregory Neven	K.U. Leuven, Belgium
Kaisa Nyberg	Helsinki Univ. of Tech. and Nokia, Finland
Carles Padró	Polytechnic University of Catalonia, Spain
Sarvar Patel	Alcatel-Lucent, USA
Sihan Qing	Chinese Academy of Sciences, China
Greg Rose	Qualcomm, USA
Rei Safavi-Naini	University of Calgary, Canada
Pierangela Samarati	University of Milan, Italy
Andre Scedrov	University of Pennsylvania, USA
Berry Schoenmakers	Technical University Eindhoven, Holland
Tom Shrimpton	Portland State University, USA
Michael Steiner	IBM Research, USA
Doug Tygar	University of California-Berkeley, USA
Wen-Guey Tzeng	National Chiao Tung University, Taiwan
Dominique Unruh	Saarland University, Germany
Ariel Waissbein	ITBA and Core Security, Argentina
Brent Waters	SRI International, USA
Susanne Wetzel	Stevens Institute of Technology, USA
Stephen Wolthusen	Royal Holloway University of London, UK
Moti Yung	Columbia University, USA
Xiaolan (Catherine) Zhang	IBM Research, USA

Hash Subcommittee

John Black	University of Colorado, USA
Shai Halevi	IBM Research, USA
Paul Hoffman	VPN Consortium, USA
John Kelsey	NIST, USA
Vlastimil Klima	Czech Republic
Stefan Lucks	Bauhaus-University Weimar, Germany
Tom Shrimpton	Portland State University, USA
Martijn Stam	EPFL, Switzerland
Ron Steinfeld	Macquarie University, Australia
Marc Stevens	Technical University Eindhoven, Holland

External Reviewers

Andre Adelsbach
Claudio Ardagna
Georges Baatz
Joonsang Baek
Billy Brumley
Matt Burnside
Reza Curtmola
George Danezis
Marie Duflot
Ratna Dutta
Sara Foresti
Ezequiel Gutesman
Martin Hirt
Susan Hohenberger
Bill Horne
Sotiris Ioannidis
Florent Jacquemard
Charanjit Jutla
Marcelo Kaihara
Darko Kirovski
Tetsutaro Kobayashi
Vladimir Kolesnikov
Yuichi Komano
Gaicheng Li
Hafiz Malik
Michael de Mare
Toshihiko Matsuo
Vishal Misra
Rossana Motta
Ginger Myles
Cedric Ng
Antonio Nicolosi
Prasad Rao
Mohammed-Reza Reyhanitabar
Mark Ryan
Siamak Shahandashti
Nicholas Sheppard
Seonghan Shin
Johan Sjoedin
Mitsuru Tada
Katsuyuki Takashima
Qiang Tang
Carmela Troncoso
Duc Liem Vo
Shabsi Walfish
Guilin Wang
Wendy Hui Wang
Qianhong Wu
Yongdong Wu
Angelika Zavou
Hong-Sheng Zhou

Sponsoring Institutions

Universidad Técnica Federico Santa María, Valparaíso, Chile
Support Center for Advanced Telecommunications Technology Research, Foundation, Japan
Microsoft Corporation
Yahoo! Research

Table of Contents

Identify-Based Cryptography

Cryptanalysis

Network Security

Software Obfuscation

Public-Key Cryptosystems

Detecting System Emulators

Thomas Raffetseder, Christopher Kruegel, and Engin Kirda*

Secure Systems Lab, Technical University of Vienna, Austria
{tr,chris,ek}@seclab.tuwien.ac.at

Abstract. Malware analysis is the process of determining the behavior and purpose of a given malware sample (such as a virus, worm, or Trojan horse). This process is a necessary step to be able to develop effective detection techniques and removal tools. Security companies typically analyze unknown malware samples using simulated system environments (such as virtual machines or emulators). The reason is that these environments ease the analysis process and provide more control over executing processes. Of course, the goal of malware authors is to make the analysis process as difficult as possible. To this end, they can equip their malware programs with checks that detect whether their code is executing in a virtual environment, and if so, adjust the program's behavior accordingly. In fact, many current malware programs already use routines to determine whether they are running in a virtualizer such as VMware.

The general belief is that system emulators (such as Qemu) are more difficult to detect than traditional virtual machines (such as VMware) because they handle all instructions in software. In this paper, we seek to answer the question whether this belief is justified. In particular, we analyze a number of possibilities to detect system emulators. Our results shows that emulation can be successfully detected, mainly because the task of perfectly emulating real hardware is complex. Furthermore, some of our tests also indicate that novel technologies that provide hardware support for virtualization (such as Intel Virtualization Technology) may not be as undetectable as previously thought.

1 Introduction

The Internet has become an integral part of our lives. Today, we interact with hundreds of services, do business online, and share information without leaving the comfort of our offices or homes. Unfortunately, the Internet has turned into a hostile environment. As the importance of online commerce and business has increased, miscreants have started shifting their focus to Internet-based scams and attacks. Such attacks are easy to perform and highly profitable. A popular technique is to develop malware (such as a Trojan horse or spyware) that is installed on victims' machines. Once deployed, the malicious software can then

* This project was supported by the Austrian Science Foundation (FWF) under grants P-18157 and P-18764, the FIT-IT project Pathfinder, and the Secure Business Austria competence center.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 1–18, 2007.

be used to capture the victims' sensitive information (such as passwords or credit card numbers) and perform illegal online financial transactions.

When an unknown malware sample is obtained by a security organization such as an anti-virus company, it has to be analyzed in depth. The goal is understand the actions the malware performs, both to devise defense and detection mechanisms as well as to estimate the damage it can inflict. To perform the analysis, running the executable in a virtual machine such as the one provided by VMware [1] is a popular choice. In this case, the malware can only affect the virtual PC and not the real one[1]. A virtual environment also has the benefit that it offers tight control over program execution, allowing the analyst to pause the system at any time and inspect the contents of the memory. In addition, the analyst can make use of snapshots that capture the state of the system at a certain point in time. This allows us to observe the effects of different actions (e.g., what happens if the malware process is killed?; what happens if a certain registry key does not exist?) without having to reinstall the system after each experiment. Instead, one can just revert back to a previously stored snapshot.

Obviously, an important question is whether a malware program can detect if it is executed in a virtual environment. If malicious code can easily detect that it is running in a simulator, it could try to thwart analysis by simply changing the way it behaves. Unfortunately, it is possible to detect the presence of virtual machines (VMs) such as VMware. In fact, a number of different mechanisms have been published [2,3] that explain how a program can detect if it is run inside a VM. These checks and similar techniques are already used by malware (e.g., [4] is using a simple detection technique). Thus, the analysis results obtained by executing malicious code inside a VM become questionable. Because of the availability of checks that can identify virtual machines, there is a general belief among security professionals that software emulation is better suited for analysis than virtualization. The reason is that an emulator does not execute machine instructions directly on the hardware, but handles them in software. Also, a number of malware analysis tools (e.g., Cobra [5] or TTAnalyze [6]) have been presented recently that claim to be stealthy (that is, undetectable by malicious code) because they are based on software emulation.

In this paper, we aim to answer the question whether software emulation is as stealthy as hoped for. Unfortunately, our results show that there are several possible methods that can be used to distinguish emulated environments from a real computer. Most of these techniques aim at identifying elements of the computer hardware that are difficult to faithfully emulate in software. In addition, we developed a number of specific checks to detect Qemu [7], a popular system emulator that forms the basis for malware analysis tool such as TTAnalyze [6]. These checks allow a program to identify observable differences in the behavior of the CPU cache, the implementation of the instruction set (such as bugs present on a particular CPU), MSRs (model-specific processor registers),

[1] Note that the software emulating the PC itself may have implementation flaws that could allow malicious code to break out of the virtual PC. However, such errors are not common.

and the I/O system. Furthermore, some of our experiments also indicate that novel technologies such as the Intel Virtualization Technology may not be as undetectable as previously thought. Because of the complexity of providing a virtualized environment that precisely mimics the real system, including its timing properties, we believe that miscreants have opportunities to identify small deviations that can be used for detection.
The contributions of this paper are as follows:

- We discuss a number of possible approaches that can be used to distinguish between an emulated environment and a real computer.
- We describe in detail specific checks that allow a program to detect the presence of Qemu, a popular system emulator that is used as the basis of malware analysis tools such as TTAnalyze.
- We examine the extent to which our emulator detection techniques apply to the novel virtualization technologies recently introduced by processor manufacturers (such as Intel VT).

2 Virtual Machine Monitors (VMMs) and Emulators

Virtual machine monitors (VMMs) and emulators are computer programs that provide mechanisms to simulate the hardware. Guest software (typically operating systems) can run within this simulated environment as if they are executed on real hardware.

2.1 Virtual Machine Monitors (VMMs)

Popek and Goldberg [8] describe a virtual machine (VM) as "an efficient, isolated duplicate of the real machine." Furthermore, they define the key characteristics of a VMM as follows:

1. The VMM provides an environment for programs, which is essentially identical to the original machine. Any program run under the VMM should exhibit an effect identical to that demonstrated if the program had been run on the original machine directly, with the possible exception of differences caused by the availability of system resources and differences caused by timing dependencies.
2. Programs that run in this environment show at worst only minor decreases in speed. A statistically significant amount of instructions need to be executed on the real hardware without interception of the VMM.
3. The VMM is in complete control of system resources. That is, it is not possible for a program running under the VMM to access resources not explicitly allocated to it. The VMM is able to regain complete control of (possibly already allocated) resources at any time.

The second point is important to be able to distinguish between VMMs and emulators. Emulators do not execute code directly on hardware without intercepting the execution (see Section 2.2). Thus, emulators typically cause a decrease in speed. VMMs on the other hand, allow execution times close to native

speed. To support virtualization, a processor must meet three requirements. These are defined by Robin and Irvine [2] as follows:

> **Requirement 1.** The method of executing non-privileged instructions must be roughly equivalent in both privileged and user mode. [...]
> **Requirement 2.** There must be a method such as a protection system or an address translation system to protect the real system and any other VMs from the active VM.
> **Requirement 3.** There must be a way to automatically signal the VMM when a VM attempts to execute a sensitive instruction. It must also be possible for the VMM to simulate the effect of the instruction.

A sensitive instruction is an instruction that, when executed by a virtual machine, would interfere with the state of the underlying virtual machine monitor. Thus, a VMM cannot allow a VM to execute any of these instructions directly on the hardware [2]. Unfortunately, it is not always possible for a VMM to recognize that a virtual machine attempts to execute a sensitive operation, especially on the Intel IA-32 architecture.

The IA-32 architecture's protection mechanism recognizes four privilege levels, numbered from 0 to 3. A higher number means lesser privileges. Usually, critical software such as the kernel of an operating system is executed with privilege level 0. Other, less critical software is executed with less privileges. Instructions that can only be executed in the most privileged level 0 are called privileged instructions. If a privileged instruction is executed with a privilege level that is not 0, a general-protection exception (#GP) is thrown [9].

Typically, VMMs run in privileged mode (privilege level 0) and a VM runs in user mode (privilege level > 0). If a VM calls a privileged instruction, it causes a general-protection exception (because the privileged instruction is executed with a privilege level greater than 0). The VMM can then catch the general-protection exception and respond to it. If all sensitive instructions have to be executed in privileged mode, a processor is considered to be *virtualizable*. That is, the VMM can catch every exception generated by the execution of sensitive instructions and emulate their proper behavior. Problems occur if there are sensitive, unprivileged instructions, as these do not cause exceptions. Unfortunately, this is the case for the IA-32 architecture, whose instruction set includes seventeen sensitive, unprivileged instructions, making the IA-32 architecture unvirtualizable [2]. This property can be exploited to detect that code is running in a virtual machine. A well-known example is the RedPill code developed by Rutkowska [3], which makes use of the SIDT Store Interrupt Descriptor Table Register.

2.2 Emulators

An emulator is a piece of software that simulates the hardware, and a CPU emulator simulates the behavior of a CPU in software. An emulator does not execute code directly on the underlying hardware. Instead, instructions are intercepted by the emulator, translated to a corresponding set of instructions for the target platform, and finally executed on the hardware. Thus, whenever a sensitive

instruction is executed (even when it is unprivileged), the system emulator is invoked and can take an appropriate action. This property makes system emulators invisible to detection routines such as RedPill, and as a result, an appealing environment for malware analysis. Unlike with virtualization, the simulated guest environment does not even have to have the same architecture as the host processor (e.g., it is possible to execute code compiled for the IA32 instruction set as a guest on a Power PC hardware or vice versa).

There are two popular techniques to implement a CPU emulator; interpretation and (variations of) dynamic translation. An interpreter reads every instruction, translates it, and finally executes it on the underlying hardware. The simplest form of a dynamic translators take one source instruction, translate it and cache it. If a instruction is cached and needed again, the translator can use the cached code. The dynamic translation technique can be extended by translating blocks of instructions instead of single instructions [10].

The big disadvantage of emulators is the incurred performance penalty. In general, every instruction (or sequence of instructions) has to be analyzed and finally executed on the underlying hardware. Even though techniques such as dynamic translation and caching can be used to speed up the execution, emulators do not reach the speed of real hardware or VMMs. However, the fact that instructions are read and interpreted makes emulators powerful tools. In particular, it enables emulators to fake instructions.

For our experiments, we used Qemu [7], an open source system emulator. The software offers a "full system emulation mode", where the processor and periphery is emulated, and a "user mode emulation", where Qemu can launch executables compiled for one CPU on another CPU [7]. In our work, we focus on the full system emulation mode. For more information about Qemu see [7] and [11]. Note that we do not consider the Qemu acceleration mode [12], which executes (most) code directly on the hardware. The reason is that in this mode, Qemu behaves similar to a classic virtual machine and no longer as a system emulator (and thus, can be detected similar to VMware).

3 General Detection Vectors

In this section, we provide an overview of possible approaches to detect emulated environments. To this end, we have to identify software whose execution differs in some observable fashion from the direct execution on hardware. In the following Section 4, we discuss concrete techniques that allow a program to determine that it is running in an emulated environment on top of Qemu.

3.1 Differences in Behavior

The first property of VMMs [8] states that a virtualized environment should behave "essentially identical to the original machine" and that "any program run under the VMM should exhibit an effect identical with that demonstrated if the program had been run on the original machine directly" (see Section 2.1).

Hence, this property should also hold for emulators. Therefore, executing the same piece of code in emulated and real environments should lead to the same results in order to have a complete and perfect emulation. If any difference can be found, one can detect emulated environments. Of particular interest are differences between the ways a real CPU and an emulated CPU behave.

CPU Bugs. Modern CPUs are complex devices. They implement the specification of a given instruction set. However, due to failures in the design and implementation, some instructions do not behave as specified. Every processor family has its typical bugs, and the presence or absence of specific bugs might allow us to distinguish between a real and an emulated CPU (assuming that the emulated CPU does not reproduce specific bugs per default). Moreover, based on the information about the processor-model-specific bugs, it might even be possible to precisely determine a specific processor model. That is, we envision the possibility to leverage CPU bugs to create processor fingerprints. To obtain information about CPU bugs, we can draw from public documentation provided by the chip manufacturers.

Model-Specific Registers. Another possible way to distinguish between the behavior of emulated and real CPUs are model-specific registers (MSRs). Model-specific registers typically handle system-related tasks and control items such as the debug extensions, the performance-monitoring counters, the machine-check architecture, and the memory type ranges (MTRRs) [9]. Note that these registers are machine-dependent. That is, certain registers are only available on specific processor models. Furthermore, the attempt to access a reserved or unimplemented MSR is specified to cause a general protection exception [13,14,15]. Thus, one can imagine a technique in which the model-specific registers of processors are enumerated to generate a specific fingerprint for different CPU models. The idea is that because of the highly system-dependent character of these registers, it might be possible that emulators do not support MSRs and, therefore, could be successfully detected. This analysis includes checking the availability of MSRs as well as interpreting their contents. A list of the MSRs of the Intel processors can be found in [15].

3.2 Differences in Timing

The performance of an application running in a virtual environment cannot be as high as on real hardware (see Section 2.1). This is because additional work such as translation or interception has to be done. Clearly, the *absolute performance* in an emulated environment is lower than that on the real hardware. However, using absolute performance values to detect virtual environments is difficult, as most computers differ with regards to hardware configurations. Performance results may even vary on the same hardware when using different operating systems and/or working loads.

Relative Performance. To address the problem that absolute performance values are unreliable when used to detect emulated environments, another approach

is to use what we call *relative performance*. With relative performance, a system is characterized by the performance ratio of two (or more) operations executed on a system. For example, one can compare the time that it takes to execute two different instructions to obtain a relative performance measure for a system. Of course, the choice of these instructions is important. Lang [16] suggested to compare the performance of accessing the control registers CR0 and CR3 with the performance of a simple NOP (no operation) instruction. Relative performance measurements can be created for emulated systems as well as for real environments. If the indicators between execution in real environments and virtual environments differ significantly, it shows that the execution times of the instructions did not change homogeneously. It is also possible to use the indicators generated on real environments as benchmarks that can later be used for comparison with indicators generated in virtual environments.

Another interesting aspect of relative performance is caching. The basic idea is to observe the effects caching has on real and emulated environments. To this end, a function is executed a number of times, and its execution time is measured. We expect that the first run will be much slower than all subsequent ones, because data and instructions will be cached after the first iteration. Then, we turn off caching and repeat the same test. This timing analysis can be used to examine the presence and effectiveness of caches. Because a processor cache is difficult to simulate, emulators may not support the CPU cache or may not support CPU cache control.

3.3 Hardware Specific Values

In simulated environments, all peripheral devices have to be emulated. These virtual devices include controllers, IDE/SCSI devices, and graphic cards. Certain emulators implement the virtual devices in a characteristic manner. Thus, various pieces of the emulated hardware are characteristic for certain emulators. Also, it might be possible to extract characteristic strings and hardware specific properties from devices. These strings include default vendor names, product information, and MAC addresses [16]. Most of today's hardware also includes monitoring software. By accessing these values in emulated environments, it is possible to determine differences to real hardware.

4 Results and Implementation Details

Based on the general detection vectors outlined above, this section provides a detailed discussion of our techniques to identify the system emulator Qemu [11]. While the results in this section are specific to Qemu, many ideas can be trivially applied to other emulators as well. Also, in some cases, we repeated tests under VMware. The goal was to determine whether a specific test to identify emulators would also apply to virtualizers.

We performed our experiments on an Intel Pentium 4 processor as well as on an Intel Core 2 Duo processor. The guest and host operating system used was

Linux Debian Testing with kernel version 2.6.16. Qemu was used with version 0.8.2, and VMware was used with version 5.5.2. We did not use the Qemu acceleration module [12] so that we could test the feature of the system emulator rather than that of the VMM.

4.1 Using CPU Bugs to Identify a Processor

This section describes our approach to fingerprint the CPU using CPU bugs. Here, we focus only on the Intel Pentium 4 processor. The document "Intel Pentium 4 Processor - Specification Update" [17] contains 104 errata, 45 of them are without a plan to fix (as of December 2006). We searched the document for errata that were not scheduled for fixing and seemed to be reproducible. Out of the 45 without a plan to fix, we selected the following two to implement:

Errata N 5: Invalid opcode 0FFFh Requires a ModRM Byte: 0FFFh is an invalid opcode that causes the processor to raise an invalid opcode exception (undefined opcode UD). This opcode does not require a ModR/M byte (the ModR/M byte is part of an instruction that immediately follows the opcode, typically specifying the instruction's addressing format). On the Pentium 4 processor, however, this opcode raises a page or limit fault (instead of the expected invalid opcode exception) when the "corresponding" ModR/M byte cannot be accessed [17].

To test for this bug, we first allocate two pages in memory. The instruction with opcode 0FFFh is put at the very end of the first page. In the next step, we use the Linux `mprotect` system call to remove all access permissions from the second page. Finally, the code at the end of the first page is invoked. On Pentium 4 processors, we receive a segmentation fault (the page fault is processed by the kernel and a segmentation fault is sent to the user mode program), because the processor attempts to load the byte following the opcode as the ModR/M byte. This fails, because the page is marked as not accessible. With Qemu and non-Intel Pentium 4 processors, the invalid opcode on the first page is sufficient to raise an exception.

Errata N 86: Incorrect Debug Exception (#DB) May Occur When a Data Breakpoint is Set on an FP Instruction: The IA-32 architecture contains extensions for debugging. These include various debug registers and debug exceptions. A complete description of the debug facilities can be found in [15]. An incorrect debug exception (#DB) may occur if a data breakpoint is placed on a floating point instruction. More precisely, the exception occurs if the floating point instruction also causes a floating point event [17].

To test for this bug, we set a data hardware breakpoint on a floating point instruction. In the next step, we have to make this instruction raise an exception. To this end, we used the FDIV (floating point division) inststruction to cause a division by zero exception. Our tests show that on Pentium 4 machines, the machine halts because the processor raises an incorrect debug exception (i.e., a data breakpoint set on an instruction should not raise a debug exception). This did not occur with Qemu or non-Pentium 4 processors.

4.2 Using Model-Specific Registers (MSRs) to Identify a Processor

According to Intel's software developer's manual [13,14,15], trying to access a reserved or unimplemented model-specific register (MSR) causes a general protection (#GP) exception. However, Qemu did not raise any exceptions when trying to access a reserved or unimplemented MSR. Hence, Qemu can be easily detected using this approach.

In our particular test, we attempt to write to the reserved MSR `IA32_MCG_RESERVED1` (number 18BH) (for list of the MSRs of the Intel processors see [15]) with the appropriate WRMSR instruction, using the following GNU C code:

```
asm volatile("WRMSR;" : : "c" (0x18b));//input: IA32_MCG_RESERVED1
```

This code, run in a kernel module (MSRs can only be accessed in privileged mode), should result in a general protection fault. Executed on real hardware (Intel Pentium 4 and Intel Core 2 Duo), it behaves as expected. The same code executed in a Qemu environment does not raise any exception.

Extending the idea above, we can generate a fingerprint for a specific processor by checking for the existence of certain MSRs. To this end, if a given MSR exists on the CPU. To test the existence of a register, we use the `rdmsr_safe` macro defined in the Linux kernel. This macro defines the assembler code for accessing a specified MSR and includes exception handling code. A user mode program accesses this kernel module using the character device. This user mode program can now extract a list of all existing MSRs. Then, it can compare this list with a database that stores the available MSRs for known processors to determine the exact model.

4.3 Instruction Length

In [9], it is stated that "the Intel processor sets a limit of 15 bytes on instruction length. The only way to violate this limit is by putting redundant prefixes before an instruction. A general-protection exception is generated if the limit on instruction length is violated." We discovered that the CPU emulated by Qemu does not set this limit.

One of the available prefixes is the Repeat String (REP) operation prefix (for a comprehensive list of prefixes see [18]). This prefix can be added to a string instruction and repeats the string instruction a number of times. If this prefix is repeated more than 14 times consecutively before a string move operation, the CPU raises an illegal instruction exception. The reason is that the instruction length limit is violated (e.g., $15 * REP(1byte) + 1 * MOVSB(1byte) > 15bytes$). However, the same instruction executed in a Qemu environment does not raise this exception. Note that the idea about the REP prefix was first posted in [19].

4.4 Alignment Checking

The IA32 architecture supports alignment checking. When code is run with the least privileges, alignment of memory addresses can be checked by the CPU.

To this end, the AM (alignment mask) flag in the CR0 (control register 0) and the AC (alignment check) flag in the EFLAGS register must be set. When alignment checking is activated, unaligned memory references generate alignment exceptions (#AC) [9]. Qemu does not support this processor feature, and thus, can be distinguished from a real processor.

To test for the existence of the alignment checking feature, we developed a kernel module that sets the alignment flag in the control register 0 (usually, this flag is set already by the operating system). Then, a user mode program sets the alignment check flag in the EFLAGS register and attempts to access an address that is both unaligned and invalid. If alignment checking is supported, an alignment exceptions (#AC) is thrown. If alignment checking is not supported, a segmentation fault occurs, because the CPU tries to access an invalid address. As expected, this code executed under Qemu results in a segmentation fault, indicating that alignment checking is not supported by Qemu.

4.5 Relative Performance – Comparison of Instructions

The goal of relative performance tests is to distinguish between real and emulated environments using timing analysis. The idea is to measure the absolute time that it takes to execute a privileged instruction and compare it to the time is takes to execute an unprivileged instruction. More precisely, the execution of an unprivileged instruction serves as the baseline. The execution time of the privileged instruction is then divided by this baseline time, yielding a relative performance number. This relative performance number is used to identify a simulated environment. The rationale is that a real processor has to perform different tasks than a simulated CPU when handling privileged instructions, and these different tasks manifest themselves by yielding different relative performance results.

For our experiment, we used a kernel module developed by Lang [16]. This kernel module reads either from the control register CR0 or from CR3, and then writes back the value previously read. To measure the time that it takes to execute instructions, the processor's time stamp counter (TSC) is used. Reading and writing to the control register CR0 (see Figure 1) is relatively faster on both VMware and Qemu than on real hardware. Interestingly, the relative timings for VMware and Qemu are very similar. Accessing the control register CR3 (see Figure 2) shows a significant timing difference between VMware and the real hardware; the real hardware is up to 100 times faster than VMware. Qemu, while significantly faster than VMware, is still slower than the real hardware.

4.6 Relative Performance – Cached Instructions

Instruction and data caches have a major influence on the performance of program execution. Thus, we were asking the question whether different timing behavior can be observed between a simulated and a real processor when executing the same piece of code for a large number of times. The idea is that caching will speed up the execution of all iterations after the first one. However,

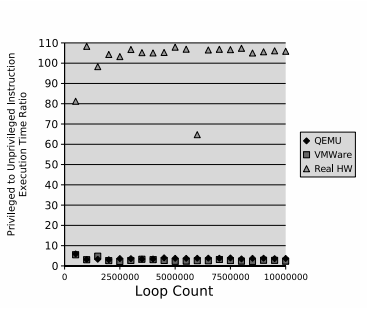


Fig. 1. Reading/Writing CR0 (TSC) on the Pentium 4 1.8 GHz

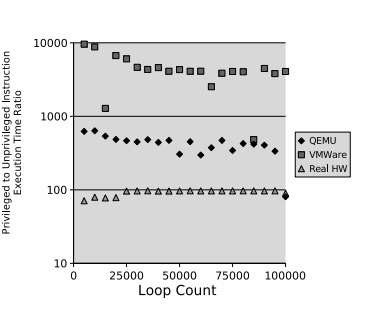


Fig. 2. Reading/Writing CR3 (TSC) on the Pentium 4 1.8 GHz

the relative speed-up could be different between depending on the environment. In addition, the timing tests were repeated when caching was disabled. Again, we wished to determine whether a different timing behavior could be seen.

For the cache timing tests, we wrote a function that contained a loop. This loop executed a sequence of simple, arithmetic instructions 100 times. For each test, this function was invoked six times (each invocation was called a run). When caching is enabled, we expect the first invocation (the first run) to last slightly longer. The reason is that the caches have to be filled. Starting from the second run, all others last shorter. To measure the execution time, the RDTSC instruction (Read Time-Stamp Counter) is used. Because the CPU supports out-of-order execution (that is, instructions are not necessarily executed in the order they appear in the assembler code), we need an instruction that forces all previous instructions to complete before we read the TSC. The CPUID instruction is one of these serializing instructions [20]. The timing code was executed in a kernel module. This allowed us to disable interrupts for each test run and therefore, to guarantee that only the timing of our test code rather than that of the interrupts handlers was measured and led to repeatable results.

After the first set of experiments with caching, we disabled caching and repeated the experiments. To disable caching, the processor has to enter the no-fill cache mode (i.e., set the CD flag in control register CR0 to 1 and the NW flag to 0) and all caches need to be flushed using the WBINVD instruction. Finally, the MTRRs need to be disabled and the default memory type has to be set to uncached or all MTRRs have to be set for the uncached memory type. [9] Turning on the cache again works the opposite way.

The charts showing the results of our experiments in this section are structured as follows: We executed six independent test runs. Each of the test runs consisted of six calls to one function. In the chart, the first bar of each test run shows the time (in cycles) of the first call, the second bar shows the time of the second call, and so on. The results are shown in Figures 3 to 8.

Two main conclusions can be distilled from the figures. One is that caching has a much more pronounced effect on the execution times when an emulated (or virtual) environment is used. The second conclusion is that for simulated environments, the timing results are the same, independent of whether caching

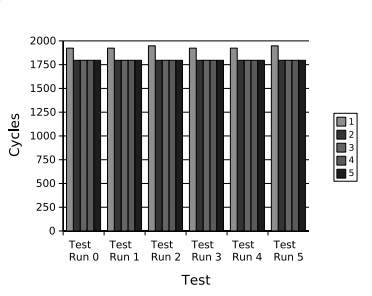


Fig. 3. Real Hardware (Cache On) - Pentium 4 1.8 GHz

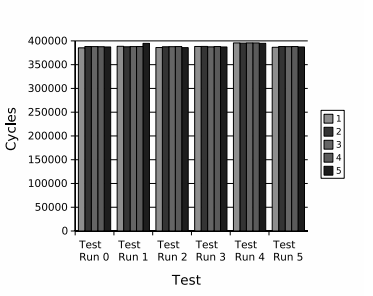


Fig. 4. Real Hardware (Cache Off) - Pentium 4 1.8 GHz

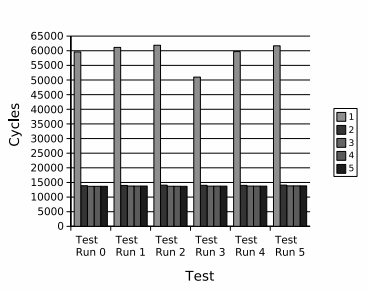


Fig. 5. Qemu (Cache On)

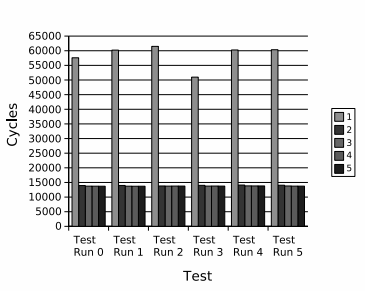


Fig. 6. Qemu (Cache Off)

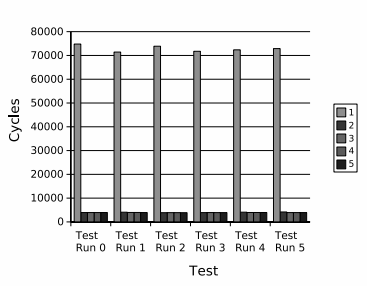


Fig. 7. VMware (Cache On)

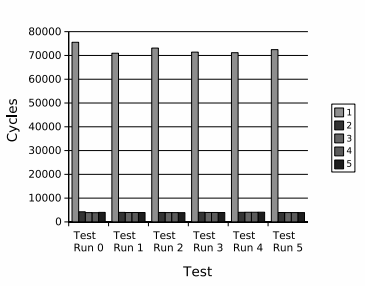


Fig. 8. VMware (Cache Off)

is active or not. In other words, both Qemu and VMware discard requests of software to disable caching. Both observations can be leveraged to distinguish between emulated and real processors.

5 Detecting Hardware-Supported VMMs

This section discusses our preliminary research in detecting virtual machine systems that make use of hardware extensions recently proposed by the two largest x86 (IA-32 architecture) processor manufacturers, Intel Corp. and Advanced

Micro Devices, Inc. (AMD). The specification of Intel's technique (called Intel VT: Intel Virtualization Technology) can be found in [21] and [15], and AMD's technique (called AMD-V: AMD Virtualization) can be found in [22] and [23]. Here, we only focus on the virtualization technique used by Intel processors.

5.1 The Intel Virtualization Technology (Intel VT)

In Section 2.1, we outlined the problems when attempting to virtualize processors that are based on the Intel IA-32 architecture. In particular, we discussed the problem of seventeen sensitive, but unprivileged instructions that make this architecture unvirtualizable. To address this problem, Intel introduced appropriate virtual-machine extensions. The key idea of these extensions is to provide a separate operation mode for VMMs. The operation mode for VMMs is called the root mode, while the mode for guest systems (virtual machines) is called the non-root mode. The code executed in root mode runs with higher privileges than all the code in the guest systems (including, of course, the guest operating system). If a guest system, even when running in processor level 0, attempts to execute instructions that would interfere with the state of other guest systems or the VMM, the processor recognizes this instruction and notifies the VMM. The VMM can then react to this instruction, and in particular, emulate the proper behavior to the guest, but limit (or prevent) its effects on the state of other virtual machines or the VMM itself. For example, moving a register to one of the control registers (e.g., CR0) would certainly interfere with the VMM and other running virtual machines. However, this instruction causes a VM exit and an immediate trap to the VMM. The VMM can then restrict the effects of the instruction and simulate its correct behavior to the guest [15].

The Intel manual [15] states that "there is no software-visible bit whose setting indicates whether a logical processor is in VMX non-root operation. This fact may allow a VMM to prevent guest software from determining that it is running in a virtual machine." This sounds promising, as it could potentially allow one to create a analysis environment that would be invisible to malware. Interestingly, it also opens up completely new possibilities for stealth malware. That is, a rootkit could install itself as a virtual machine monitor and virtualize the user's operating system. In theory, such a rootkit could not be detected by conventional methods. Prototypes of this type of malware are the well-known BluePill [24], Vitriol [25], or SubVirt [26]. Unfortunately, despite our requests, we were not able to get access to either Blue Pill or Vitriol, and thus, could not test our detection techniques with these rootkits. Instead, we had to focus on the widely used VMware Workstation 5.5.2 [1], which has support for Intel VT, and the open source program KVM, a kernel-based virtual machine for Linux that also leverages the new Intel extension [27].

Timing Analysis. In Section 4.6, we presented a timing analysis that used cache effects to allow us to distinguish between a system emulator and a real machine. Recall that system emulators and VMMs use the cache provided by the underlying hardware. Thus, their cache behavior is very similar to that of

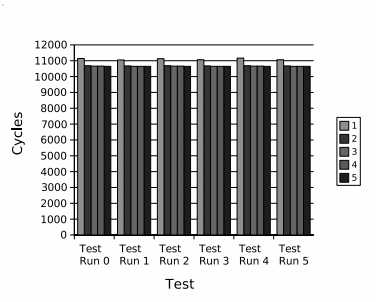


Fig. 9. KVM (IVT On - Cache On)

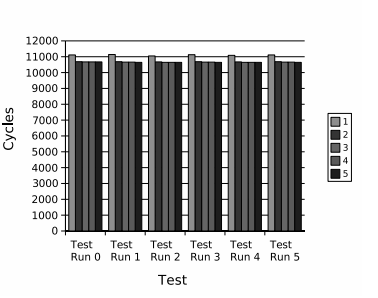


Fig. 10. KVM (IVT On - Cache Off)

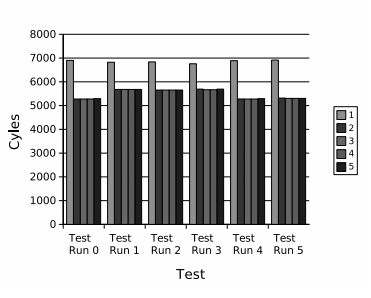


Fig. 11. VMware (IVT On - Cache On)

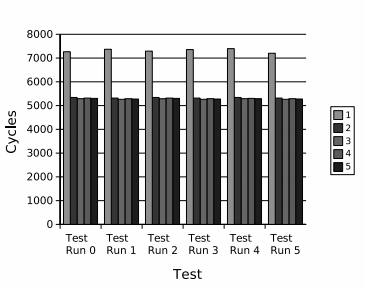


Fig. 12. VMware (IVT On - Cache Off)

real hardware. As expected, code and data that has been accessed recently is delivered much faster when requested again. Interestingly, however, disabling the cache in the virtual environment does not lead to any changes in the observed behavior of the processor. That is, the virtual machine and the system emulator behaved *identically* independent of whether caching was enabled or disabled. Of course, timing results show that execution is significantly slower on a real machine when caching is disabled.

The discussion above shows that system emulators and traditional VMMs do not allow the guest system to disable the cache. This provides a straightforward mechanism for a program to determine whether it is running in a virtual environment. The question is whether the detection approach also succeeds when the VMM uses the Intel VT hardware extensions. The results of our analysis are shown in Figures 9 to 12 (The results of the test runs using the Core 2 Duo processor, without any virtualizers or IVT enabled, are comparable to the results on the Pentium 4 processor (see Figures 3, 4, 7 and 8) and therefore not listed here again.) It can be clearly seen that our cache timing analysis has the ability of distinguishing between a virtual environment and real hardware, even when Intel VT is used. Of course, a VMM that is aware of our timing tests might attempt to adjust its behavior such that it appears as real hardware. For example, we use time stamp counters (TSC) to measure the cycles that our timing program is executing. Thus, the VMM could modify the value of the TSC appropriately

whenever we access it (using an instruction that traps into the VMM). Nevertheless, even though the TSC can be set to arbitrary values, the VMM would have to adjust the value depending on the number and the type of executed instructions. This is a non-trivial task, as any deviation from the expected result would serve as an indication that the test code is running in a virtual environment.

Also, turning off the cache involves setting a certain bit in the control register 0 (CR0), invalidating the cache (using the WBINVD instruction), and writing to a certain MSR (using the WRMSRS instruction). All these instructions cause VMX exit transitions. and it would be possible for the VMM to simply disable caching on the underlying processor. However, this would slow down the VMM as well as all other running guest systems executed on the processor. This is certainly not practical for VMMs in production settings (i.e., a program that provides an *isolated* duplicate of the real machine). On the other hand, a rootkit could simply turn off caching, as its main goal is to remain hidden.

6 Related Work

Malicious code is a significant security problem on today's Internet. Thus, a large body of previous work deals with different solutions to detect and analyze malware. In general, these solutions can be divided into two groups: *static analysis* and *dynamic analysis* techniques.

Static analysis is the process of analyzing a program's code without actually executing it. This approach has the advantage that one can cover the entire code and thus, possibly capture the complete program behavior. A number of static binary analysis techniques [28,29,30] have been introduced to detect different types of malware. The main weakness of static analysis is the fact that the code analyzed may not necessarily be the code that is actually run. In particular, this is true when the malware makes use of code obfuscation.

Because of the many ways in which code can be obfuscated and the fundamental limits in what can be decided statically, dynamic analysis is an important and popular alternative. Dynamic techniques analyze the code during run-time. While these techniques are non-exhaustive, they have the significant advantage that only those instructions are analyzed that the code actually executes. Of course, running malware directly on the analyst's computer is not possible, as the malicious code could easily escape and infect other machines. Furthermore, the use of a dedicated stand-alone machine requires to reinstall the system after every test. Thus, the question arises in which environment a malware sample should be executed.

Virtual machines such as VMware [1] are a common and comfortable choice as an environment for malware analysis. Unfortunately, malicious code has adapted, and malware authors increasingly include routines into their creations that check for the presence of such virtual machines. These checks are actually quite straightforward, as the Intel IA-32 instruction set contains a number of instructions that are unvirtualizable [2]. Based on these instructions (and a number of alternative methods), a variety of detection techniques have been implemented.

This includes RedPill [3], but there are also other VMware fingerprinting suites such as "Scooby Doo - VMware Fingerprint Suite" [31]. Recent publications also deal with general virtualization anomalies [32] and the usage of timing benchmarks to detect the presence of VMMs [33].

To address the problem of analysis environments that are easy to detect by malware, researchers have proposed systems based on emulation. For example, Cobra [5] and TTAnalyze [6] are two malware analysis tools that are based on full system emulation. While there was a general belief that these system are stealthier (i.e., less easy to detect), the extent to which this is true was unclear. In this paper, we analyzed possible approaches that can be used to identify system emulators. Moreover, we presented a number of concrete checks that allow us to detect Qemu (the emulator that TTAnalyze [6] is based on). The work closest to ours is a recent paper by Ferrie [34]. In his paper, Ferrie demonstrates mechanisms to detect a large number of virtual machine monitors, but he also addresses the issue of identifying emulators. The difference to this work is that we present a variety of approaches to detect system emulators, while Ferrie only focuses on a few specific implementation bugs of system emulators that incorrectly reproduce the behavior of specific instructions.

Finally, there has been recent work by processor vendors such as Intel and AMD to provide hardware extensions that help to make a virtual environment undetectable. While clearly a step into the right direction, there are properties (especially timing related ones) that are difficult to model precisely in a virtual environment and thus, provide malicious code with the possibility for detection (as our preliminary results indicate).

7 Conclusion

In this paper, we presented a number of techniques that can be used to detect system emulators. These techniques make use of specific CPU bugs, model-specific registers (MSRs), instruction length limits, alignment checking, relative performance measurements, and specific hardware and I/O features that are difficult to emulate. The conclusion that we have to draw from our experiments is that emulators are not necessarily stealthier than virtual machines. It is theoretically possible to adapt a system emulator to address each of the specific detection methods that we have outlined above. However, it is still an arms race as the adversary can find new indicators that are not covered yet. The underlying problem is that virtual machines and system emulators are not written with malicious code analysis in mind. Thus, it is typically enough to provide a simulated environment that is sufficiently close to the real system. Unfortunately, as far as security analysis is concerned, the emulation has to be perfect. Otherwise, the malware under analysis can discover that it is not running in a real environment and hence, adapt its behavior. Our experiments also indicate that some of our tests might be applicable to detect new virtualization technologies recently introduced by Intel and AMD, making the task of creating a perfectly stealth analysis environment very difficult.

Given the discussion in this paper, it may seem as if security researchers are losing the fight against malware authors with respect to hiding their analysis infrastructure. However, note that virtualization and system emulation are increasingly gaining popularity among a broad spectrum of computer users. These environments are not only useful for malware analysis, but also make system maintenance and deployment much easier. As a result, malicious code cannot expect any longer that a virtual environment is an indication of an analyst examining the code. Instead, it could also be a production server that uses virtualization technology.

References

1. VMware Inc. (2006), http://www.vmware.com/
2. Robin, J.S., Irvine, C.E.: Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. In: Proceedings of the 9th USENIX Security Symposium, Denver, Colorado, USA (August 14–17, 2000)
3. Rutkowska, J.: Red Pill... or how to detect VMM using (almost) one CPU instruction (2004), http://invisiblethings.org/papers/redpill.html
4. Classified by Symantec Corporation: W32.Toxbot.C (2007), http://www.symantec.com/security_response/writeup.jsp?docid=2005-063015-3130-99&tabid=2
5. Vasudevan, A., Yerraballi, R.: Cobra: Fine-grained Malware Analysis using Stealth Localized-Executions. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (2006)
6. Bayer, U., Kruegel, C., Kirda, E.: TTAnalyze: A Tool for Analyzing Malware. In: 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR) (2006)
7. Qemu - open source processor emulator (2006), http://fabrice.bellard.free.fr/qemu/
8. Popek, G.J., Goldberg, R.P.: Formal requirements for virtualizable third generation architectures. Communications of the ACM 17(7), 412–421 (1974)
9. Intel Corporation: Intel 64 and IA-32 Architectures Software Developer's Manual vol. 3A: System Programming Guide, Part 1 (2006)
10. May, C.: Mimic: a fast system/370 simulator. In: Conference on Programming Language Design and Implementation - Papers of the Symposium on Interpreters and interpretive techniques (1987)
11. Bellard, F.: Qemu, a Fast and Portable Dynamic Translator. In: USENIX 2005 Annual Technical Conference, FREENIX (2005)
12. Bellard, F.: Qemu Accelerator Module (2006), http://fabrice.bellard.free.fr/qemu/qemu-accel.html
13. Intel Corporation: Intel 64 and IA-32 Architectures Software Developer's Manual vol. 1: Basic Architecture (2006)
14. Intel Corporation: Intel 64 and IA-32 Architectures Software Developer's Manual vol. 2B: Instruction Set Reference, N-Z (2006)
15. Intel Corporation: Intel 64 and IA-32 Architectures Software Developer's Manual vol. 3B: System Programming Guide, Part 2 (2006)
16. Lang, J.: Personal Correspondence (2006)
17. Intel Corporation: Intel Pentium 4 Processor - Specification Update (2006)
18. Intel Corporation: Intel 64 and IA-32 Architectures Software Developer's Manual vol. 2A: Instruction Set Reference, A-M (2006)

19. VirtualPC 2004 (build 528) detection (?) (2006), http://www.securityfocus.com/archive/1/445189
20. Intel Corporation: Using the RDTSC Instruction for Performance Monitoring (1997)
21. Intel Corporation: Intel Virtualization Technology Specification for the IA-32 Intel Architecture (2005)
22. Advanced Micro Devices, Inc.: AMD64 Architecture Programmer's Manual vol. 2: System Programming (2006)
23. Advanced Micro Devices, Inc.: AMD I/O Virtualization Technology (IOMMU) Specification (2006)
24. Rutkowska, J.: Introducing Blue Pill (2006), http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.ht ml
25. Zovi, D.D.: Hardware Virtualization Rootkits. In: BlackHat Briefings, USA (2006)
26. King, S.T., Chen, P.M., Wang, Y.M., Verbowski, C., Wang, H.J., Lorch, J.R.: SubVirt: Implementing malware with virtual machines. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (2006)
27. KVM: Kernel-based Virtual Machine (2007), http://kvm.sourceforge.net/
28. Christodorescu, M., Jha, S.: Static Analysis of Executables to Detect Malicious Patterns. In: Usenix Security Symposium (2003)
29. Christodorescu, M., Jha, S., Seshia, S., Song, D., Bryant, R.: Semantics-aware Malware Detection. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (2005)
30. Kruegel, C., Robertson, W., Vigna, G.: Detecting Kernel-Level Rootkits Through Binary Analysis. In: Yew, P.-C., Xue, J. (eds.) ACSAC 2004. LNCS, vol. 3189, Springer, Heidelberg (2004)
31. Klein, T.: Scooby Doo - VMware Fingerprint Suite (2006), http://www.trapkit.de/research/vmm/scoopydoo/index.html
32. Garfinkel, T., Adams, K., Warfield, A., Franklin, J.: Compatibility is Not Transparency: VMM Detection Myths and Realities. In: Proceedings of the 11th Workshop on Hot Topics in Operating Systems (HotOS-XI) (May 2007)
33. Franklin, J., Luk, M., McCune, J.M., Seshadri, A., Perrig, A., van Doorn, L.: Remote Detection of Virtual Machine Monitors with Fuzzy Benchmarking. Carnegie Mellon CyLab (2007)
34. Ferrie, P.: Attacks on Virtual Machine Emulators. In: AVAR Conference, Auckland, Symantec Advanced Threat Research (2006)

Features vs. Attacks: A Comprehensive Feature Selection Model for Network Based Intrusion Detection Systems

Iosif-Viorel Onut and Ali A. Ghorbani

Information Security Centre of Excellence,
Faculty of Computer Science, University of New Brunswick,
Fredericton NB E3B 5A3, Canada
{onut.viorel,ghorbani}@unb.ca
http://nsl.cs.unb.ca

Abstract. One of the most crucial development phases of a network intrusion detection system is the feature selection one. A poorly chosen set of features may lead to a significant drop in the detection rate, regardless of the employed detection method. Despite its importance, we believe, that this research area lacks of comprehensive studies. Our research proposes a model for mining the best features that can be extracted directly from the network packets, by ranking them against their statistical properties during the normal and intrusive stages. As proof of concept, we study the performance of 673 network features while considering a set of 180 different tuning parameters. The main contribution of this work is that it proposes a ranking mechanism to evaluate the effectiveness of features against different types of attacks, and that it suggests a pool of features that could be used to improve the detection process.

Keywords: Feature evaluation, Network Intrusion Detection.

1 Introduction

One of the first tasks in the implementation of an Intrusion Detection System (IDS) is the feature selection phase. This step will definitely influence the performance of any detection engine, regardless of the technique that the engine uses. While we acknowledge that the detection technique is a decisive factor of the detection process, we also believe that the type of features that a technique uses plays a heavy role too. Despite its importance, to our knowledge, there appears to be a lack of research in this domain.

The main aim of this work is to propose a statistical technique for highlighting the most important features to be used in the detection process. The current work statistically evaluates 673 network features that are extracted directly from the network packets. The computed features are defined for the most used protocols at the Transport, Network, and Network Access layers of the TCP/IP Architecture Model (i.e., IP, ICMP, TCP, and UDP). The evaluation process considers

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 19–36, 2007.

the behavior of the features during the attack and normal phases. However, the proposed model could be successfully applied on higher layer features that can be extracted from the packet payloads.

Our model also considers the tuning of each feature as an decisive factor that influence the usefulness of that particular feature. To address this issue, we have chosen a set of 180 different tuning combinations for each feature. The tuning parameters are chosen so that they cover a fair range of possible tunings.

Our proof of concept experiments are done on the DARPA Intrusion Detection and Evaluation Dataset [2]. This dataset still remains the primary source of many papers due to the labels that it provides, the diversity of attacks that it has, the amount of captured data, and lack of other more recent available labeled datasets. In order to process the high number of feature-tuning combinations, the experimental results were conducted on a Sun V60 computer cluster provided by our university.

The rest of the paper is organized as follows, Sec. 2 summarizes the state of our research and related research in this direction of study. Next, Sec. 3 presents the logics behind the feature tunings that we used. The feature extraction process is explained in Sec. 4, followed by our statistical evaluation technique for feature relevancy explained in Sec. 5. The experimental results are presented in Sec. 6. Finally, the conclusion and possible research directions are specified in the last section of the work.

2 Background Review

One of the main challenges when dealing with the amount of information that can be extracted directly from network packets is to create a set of features that covers most of the information space. The network features are constructed around the main abstractions of the network security domain such as: packet, connection, host, and network. The idea of constructing features that will cover a reasonable part of the information space is especially hard due to the diversity of data that passes through a network link and the lack of unanimously accepted network feature classification schema in the research community. Furthermore, researchers have empirically demonstrated that false correlations between the features that may be extracted by an IDS/IPS can lead to purer results concerning the accuracy and performance of the detection system [13,12]. For example, if only 17 carefully selected features are used among all 41 features provided in the International Knowledge Discovery and Data Mining Competition (KDD-1999) [1], the detection rate will not change, but the speed of the detection algorithm will improve by about 50% [12].

There seems to be at least two main types of features that are widely accepted in the literature. This distinction is done between features that are computed based on a single TCP connection and features that are computed based on multiple TCP connections. However, the names of the previously mentioned categories are different from researcher to researcher. Accordingly, the single TCP connection features are referred to as *Basic Features* in [4,5], *Essential*

Attributes in [15], *Basic Features of individual TCP connection* in [1], *Basic TCP Features* in [10]. A superset of this feature category that also includes connectionless protocols is mentioned in [11] as *Flow Statistics* and in [14] as *Single Connection Dependent Features*. Similarly, different names are used for the second feature category too, such as *Derived Features* in [4,5], *Traffic Features* in [15,1], and *Multiple Connection Features* in [14]. Furthermore, quite a few works also recognize that there are two main ways to construct those features that target interrelationships between different flows as follows: *Time Based Features* and *Connection Based Features* [5,4,15,1,14]. The idea behind their construction is to use a sliding window over the captured data that is used to observe and extract their value. In the case of *Time Based Feature* category the sliding window specifies a predefined amount of time, while in the case of the *Connection Based Feature* category the sliding window specifies the number of connections that are considered when computing the features.

We recently proposed a new feature classification schema for network intrusion detection, which allowed us to tackle various aspects that can be extracted at network level[14]. The proposed schema contains 27 categories that, we believe, semantically cover all the main feature types that can be extracted from the network. The detail of this feature classification schema and the reasoning behind its categories is beyond the scope of this paper; however, we have used this schema to construct all the 673 features that are analyzed in this study.

3 Feature Tunings

The current work focuses on the performance of each individual feature regardless of the feature category that it belongs to. However, different features have different tuning parameters. Thus, for identifying the set of parameters that needs to be tuned, the features are grouped based on their underlying implementation, not based on their semantic definition and meaning. Consequently, we have *basic features*, *time based features*, and *connection based features* as primary types of features that we will work with.

Each of the previous groups has different tuning parameters as described in this section. These parameters heavily influence the effectiveness of each feature in detecting attacks. Thus, it is of great importance to study multiple tuning values for each feature while evaluating its performance against attacks.

The *Basic Features* category consists of all features that can be extracted from a single packet without requiring any kind of extra information. The feature candidates for this category can be any field of the datagram such as protocol, source and destination ports, flags, ICMP Type, to name a few. Extracting these features is extremely easy and fast since the feature constructor needs to examine only a single packet at a time. No extra dependency information is required, and thus, no need for tuning in this case. Even though this category is the easiest type of features that can be created, it is also the most inefficient one to use (see Sec. 6). The same cannot be said regarding the time and connection based features. Each type of feature depends on several factors that naturally lead

to different values for the same feature(s). As explained in Sec. 5, in order to evaluate each feature we consider a set of tunings that will allow us to decide if that particular feature is or is not reliable in attack detection. The logics behind the tunings that we work with are explained in the next two subsections.

3.1 Time Based Features

There are two main decisions that must be made at the creation of a time interval (i.e., *window size* and *window granularity-step*). These two settings are especially important since they both influence the speed of the feature extraction process as well as the size of the memory that will be used. Moreover, theoretically speaking there is a common expectation that different tunings will detect different types of attacks (e.g., Bursty vs. Stealthy Attacks).

The *window size* tuning is self explanatory; it is the size of the time window interval that is considered when computing this type of features. Common values for this tuning factor vary from researcher to researcher. Furthermore, an exact clear cut for the best value is hard to be made especially as this value is network dependent. Our experiments consider various values of the *time window size* in the range of one second and one hour. While we acknowledge that this interval restriction limits the number of tunings that are studied, we also argue that to the best of our knowledge most of the window sizes implemented in real systems are less than several minutes and they don't even come close to the one hour interval size. Moreover, as the window size increases the memory consumption increases too since the behavior of each feature for the past time interval needs to be stored. We have chosen 30 distinct *time window sizes*. Since the maximum time interval is roughly one hour (i.e., 55 minutes and 36 seconds) we could have linearly split the interval into 30 equal slices and consider those splitting points as tuning parameters. However, we believe that an exponential based splitting is more suitable in this case. Consequently, 33% of the tuning factors are within the first 5 minutes, the next 33% tunings are between 6 and 20 minutes, while the last 33% covers the 20 minutes to 1 hour interval. This is also justifiable since as the time interval increases the differences between the number of seconds within a time interval and the number of seconds between two consecutive intervals becomes more prominent.

The following formula gives the exact tuning values that we used in our experiments:

$$\tau_{\text{tw}}(x) = \lceil 210 * 1.099^{x} - 230 \rceil \qquad | \qquad \forall x \in [1, 30], x \subset N^{*} \tag{1}$$

where, x is a natural number denoting the x^{th} tuning sample, and τ_{tw} is the actual value in seconds for the *time window size*.

The second factor (i.e., *window granularity-step*), represents the time latency between two consecutive updates on the time interval. While this should ideally be as small as possible, in practice having to update a *time interval* (e.g., one millisecond) is not desirable, since it will overwhelm the processor. On the other hand, a large granularity will save a lot of processing power, but will introduce

delays between the actual and the computed values. For instance, a one minute granularity will introduce a one minute delay between the actual value that should have been computed and the computed one. Consequently, the longer the granularity is the sparser will be two consecutive values of the same feature, but the faster will be the processing time. In order to experience the effect of different granularity values for the *time window interval* we consider 5 different granularity values as follows:

$$\tau_{\mathrm{g}} = \{1, 2, 4, 5, 10, 20\} \tag{2}$$

where, τ_{g} is the actual value in seconds for the *time window granularity*. We stopped our analysis at 20 seconds granularity since increasing it further would basically lead to poorer results. Also, this will not serve our purpose, since based on our experiments as granularity increases, the system becomes unable in detecting fast attacks, and late in detecting the stealthy ones. As a general tuning suggestion concerning the *granularity* factor, we believe that the smaller this value is, the closer the time features are to the real values, the better the chances of catching an attack are, and the faster the detection will be. Thus, we have chosen most of the granularity values close to one second.

Consequently, for each *time feature* we study, the system will generate 180 tuning values, which represent the combination of 30 different *time window sizes* with 6 different *granularities*. Note that we ignore all tuning combinations (i.e., a total of five) that have the *time window size* smaller than the time *granularity*, since we consider those to be meaningless.

3.2 Connection Based Features

Constructing *connection based features* is of great importance since those features are built based on comparisons among different connections. A connection can be generally defined as the act of exchanging information between two hosts while using the same source and destination ports (if applicable) and the same protocol. As stated, this definition applies not only to connection oriented protocols (e.g., TCP) but also to connectionless protocols (e.g., UDP, and ICMP).

There are two main factors that influence the final value of a feature from this category. The first factor is the *window size* of the connection interval. This size specifies the number of recent connections that are considered when computing a *connection based feature*. Ideally the size of the connection interval would be sufficient enough to extract meaningful data from the network. However, the longer this connection interval is, the more memory is used to store the past connections, and the more computational time is needed. Furthermore, we believe, that finding a "suitable" connection window size is a difficult task, especially because this differs from network to network and from throughput to throughput. Consequently, our experiments are run on 30 different *connection window sizes* that vary from a size of 3 to approximately 200. We choose this range because it covers a reasonable interval of possible sizes. To the best of our knowledge, the largest size interval we encountered in other works was of 100

connections, but most of the works use even smaller sizes. The 3 to 200 interval is exponentially divided into 30 distinct slices, by following the same technique as in the *time window based* case. Thus, 33% of the tuning values range between 3 and 15 connections, the next 33% are up to 60 connections, wile the last 33% are within 60 and 200 connections.

The following formula gives the exact tuning values that we used in our experiments:

$$\tau_{\text{cw}}(x) = \lceil 5 * 1.13^x - 3 \rceil \qquad | \qquad \forall x \in [1, 30], x \subset N^* \tag{3}$$

where, x is a natural number denoting the x^{th} tuning sample, and τ_{cw} is the actual value in number of connections for the *connection window size*.

The second tuning factor for the *connection based features* emerged as a consequence to the definition of a connection. Recall that a connection is defined as the act of exchanging information between two hosts. While determining the beginning and ending of a connection implemented through a connection based protocol (such as TCP) is straight forward, the same cannot be said about a connectionless protocol (such as UDP and ICMP). In this case, the beginning of the connection could be marked as the first packet that is exchanged between the two hosts (considering also the port pair for UDP); however, determining the end of the connection is not a trivial task anymore. Consequently, in this case we consider the lack of activity (i.e., packets exchanged) between the two hosts as a primary indicator for determining the closing of a connection. Let us define the *connection time to live* interval as the time interval between the last packet exchanged in a connection and the end of that connection. To be consistent, we apply this mechanism to connection based protocols too. Thus, once no packets are exchanged for a period longer than the *connection time to live* interval, the connection will be automatically closed. As a side effect to this restriction, the memory is also efficiently utilize. This is especially true in the case of an on-line system where the efficient resource use is a must, and where the system does not have the luxury of maintaining an inactive connection forever. In our experiments we used 6 different *connection time to live* values that range from 20 seconds to approximately 30 minutes and are defined using the following equation:

$$\tau_{\text{ttl}}(x) = \lceil 15 * 2.209^x - 20 \rceil \qquad | \qquad \forall x \in [1, 6], x \subset N^* \tag{4}$$

where, x is a natural number denoting the x^{th} tuning sample, and τ_{ttl} is the actual value in seconds for the *connection time to live* parameter.

Conclusively, for each *connection feature* that is studied, the system will have 180 tuning values, which represents the combination of 30 different *connection window sizes* with 6 different *time to live* values.

4 The Feature Extraction Process

This section presents the main components of the *Feature Extraction Module* that we have implemented. The aim of the design was to have a highly customizable

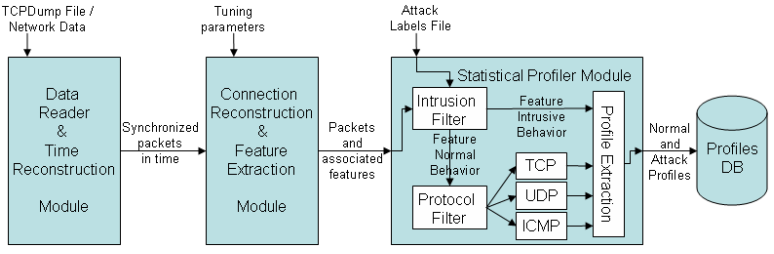


Fig. 1. The underlying architecture of the feature extraction module

module, easily extendable, that would allow the extraction of all features that we previously proposed [14], and would work either on-line or off-line. Figure 1 depicts the high level view of the feature extraction process. The first module is the *Data Reader & Time Reconstruction Module*. It is in charge of collecting data by either sniffing it from a network connection, or extracting it from an already saved tcpdump file. The *Time Reconstruction* functionality is needed only in the case of reading from a tcpdump file since the packet inter-arrival times need to be considered. This is especially important since the inter-arrival times directly influence the feature extraction process. To speedup the extraction process, the replay speed can also be adjusted.

Once this step is done, the synchronized packets are sent to the *Connection Reconstruction and Feature Extraction Module*. This module is in charge of constructing and storing the encountered connections, and of creating the *Time Based Features* and the *Connection Based Features*. It receives as input the synchronized packets and the current tuning parameters, and it computes the associated features. The detail description of this module is beyond the scope of this paper; however, further details can be found our previous work [14].

The final step of the feature extraction process is the attack and normal profile generation. The task is accomplished by the *Statistical Profiler Module*. This module uses the provided attack labels to filter out the intrusive from the normal behavior of each individual feature, and stores the corresponding statistical data into uniquely identifiable profiles. Each profile keeps track of the mean μ and standard deviation σ statistics of a particular feature during the normal or intrusive behavior. It is known that the features tend to have different values for different protocols. For instance, the size of the ICMP packets is expected to be smaller than the size of the TCP packets. Thus, instead of creating a single profile for the normal behavior of a feature, our system creates individual normal profiles for each protocol that applies to the current feature. Let f_i represent the i^{th} feature to be analyzed, ξ_j denote the j^{th} encountered intrusion, and τ_k represent the k^{th} tuning factor selected. Consequently, two types of profiles are defined as follows:

– Normal profile: A normal profile is uniquely identified by a $< f_i, \tau_k >$ tuple, and characterizes a particular feature f_i extracted using τ_k tuning during the normal network operation. The profile keeps track of the $\mu(f_i, \tau_k)$ and

$\sigma(f_i, \tau_k)$ statistics representing the normal mean and standard deviation during the previously specified scenario.

– Intrusive profile: An intrusive profile is uniquely identified by a $< f_i, \xi_j, \tau_k >$ tuple, and characterizes a particular feature f_i extracted using τ_k tuning while under the ξ_j attack . Similarly, the profile keeps track of the $\mu(f_i, \xi_j, \tau_k)$ and $\sigma(f_i, \xi_j, \tau_k)$ statistics representing the mean and standard deviation of the intrusive behavior.

All these statistical profiles are stored in the *DB Profiles* database, and are constantly updated. Note that the τ_k factor depends on the type of the current feature f_i that is analyzed as explained in Sec. 3.

The process of extracting profiles is repeated once for each tcpdump file and tuning combination, until all the possible combinations are exhausted. In the current experiment the number of combinations was 3420 as explained later in Sec. 6. The implementation of this module is done as a combination of C++ and Java languages (i.e., everything except the *Profile Extraction Task* is implemented in C++), and is compiled/executed under Linux OS. Once all the profiles are created, the *Statistical Extractor Module* is invoked as described in the next section.

5 The Feature Statistical Ranking

The feature statistical ranking process is designed as a 3-tier *Statistical Extractor Module*, as depicted in Fig. 2. The module is implemented as a combination of Java and Matlab languages under Windows OS. Once all the profiles are computed for all the tuning factors and all the input data, the *Statistical Extractor Module* is invoked. One of the particularities of this module is that each tier needs to wait for the previous one to finish its job. Consequently we have introduced temporary databases between consecutive tiers, so that the intermediary data is saved. The ultimate goal of this module is to compute the probability of each individual feature f_i to be able to detect one of the main types of attacks defined in the dataset (i.e., Denial of Service (DoS), Probing, Remote to Local (R2L), and User to Root (U2R) attacks as explained in the next section). Thus, let us define $P(f_i|\text{DoS})$, $P(f_i|\text{Probe})$, $P(f_i|\text{R2L})$, $P(f_i|\text{U2R})$ as the probability of f_i to be effective in the detection of DoS, Probe, R2L and U2R attacks, respectively. The higher a probability is the better the attack detection will be. Once all of

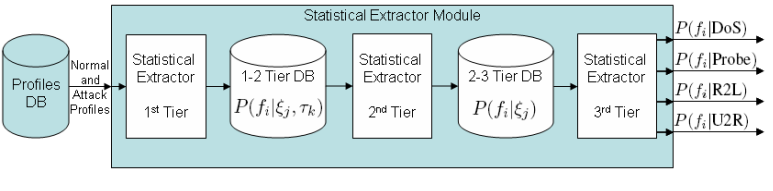


Fig. 2. The underlying architecture of the Statistical Extractor Module

these probabilities are computed, the features are sorted by their effectiveness (i.e., probability) and reported in a table like structure as listed in Sec. 6. This final step is not illustrated in Fig. 2 due to its simplicity. The following three subsections describe in detail each of the three tiers that exist in the *Statistical Extractor Module*.

5.1 Statistical Extractor 1^{st} Tier

The aim of this module is to compute the probability of a certain feature f_i in detecting a given attack ξ_j when that feature is created using the τ_k tuning factor. To do this, the current subcomponent searches the *Profiles DB* and extracts the corresponding normal and abnormal profiles.

Each profile can be pictured as one statistical distribution created by that particular feature. Furthermore, each profile keeps the μ and σ of that distribution. Our basic assumption is that f_i is influenced by ξ_j if its value significantly changes from the normal one during the attack period. Consequently, this change will naturally lead to a change in the features mean and standard deviation. Thus, we would expect that the statistical distribution during the normal behavior to be significantly different from the one during the intrusive behavior. The more different those two distributions are, the better are the chances for that feature to detect the intrusive behavior.

To study the statistical nature of a single distribution, we use *The Chebyshev's Inequality*, a well known formula named after the Russian mathematician L.P. Chebyshev who first proved it. The formula mathematically proves that in any probability distribution nearly all values are close to the mean value [16,6]. The inequality is mathematically defined as:

$$P(|X - \mu| \geq m\sigma) \leq \frac{1}{m^2}, \qquad \forall m, m \subset \mathbb{R}^* \tag{5}$$

where X is a random variable, with an expected μ value and a finite standard deviation σ. The theorem states that the probability of a value X to be more than $m\sigma$ apart from the μ is less or equal than $1/m^2$. Despite its loose bounds the theorem is very useful because it applies to any kind of distribution. Once the μ and σ are known for a distribution, the m factor can simply be computed as:

$$m = \frac{|X - \mu|}{\sigma} \tag{6}$$

It is easily seen that when $m \to \infty$, the $P(|X - \mu| \geq m\sigma) \to 0$; meaning that the farther the X point is from the current distribution the less probable it is for that point to belong to the current distribution.

Let us define $P(X|\mu, \sigma)$ as the probability of a point X to be an outlier of a distribution that has μ as mean and σ as standard deviation as follows:

$$P(X|\mu, \sigma) = \begin{cases} 1 - P(|X - \mu| \geq m\sigma) & \text{if } |X - \mu| \geq \sigma \\ 0 & \text{otherwise} \end{cases} \forall X \tag{7}$$

The formula has the same value (i.e., zero) for all the points that are within one σ away from the μ. This is easily acceptable since the probability of a point to belong to a certain distribution is very high if it is situated within one σ from the μ (i.e., about 68.26% of the population lies within 1 standard deviation based on the *Empirical Rule* and *Central Limit Theorem* [6]).

To evaluate the degree of effectiveness of a feature in detecting a certain intrusion, two important factors must be considered as follows: (a) the probability of the normal points to be outliers of the attack data distribution, and (b) the probability of the intrusive points to be outliers of the normal data distribution. This two-way check is extremely important since it considers both normal and intrusive distributions of the feature.

Let $P(f_i|\xi_j, \tau_k)$ represent the probability of feature f_i to detect intrusion ξ_j given the τ_k tuning. Recall that for each f_i, ξ_j, τ_k combination the $\mu(f_i, \tau_k)$ and $\sigma(f_i, \tau_k)$ of the normal behavior as well as $\mu(f_i, \xi_j, \tau_k)$ and $\sigma(f_i, \xi_j, \tau_k)$ of the intrusive one are available. Thus, the previous probability is defined as follows:

$$P(f_i|\xi_j, \tau_k) = \frac{P(\mu(f_i, \xi_j, \tau_k)|\mu(f_i, \tau_k), \sigma(f_i, \tau_k)) + P(\mu(f_i, \tau_k)|\mu(f_i, \xi_j, \tau_k), \sigma(f_i, \xi_j, \tau_k))}{2} \quad (8)$$

where, $P(\mu(f_i, \xi_j, \tau_k)|\mu(f_i, \tau_k), \sigma(f_i, \tau_k))$ is the probability of intrusion's mean to be an outlier of the normal data distribution, and $P(\mu(f_i, \tau_k)|\mu(f_i, \xi_j, \tau_k), \sigma(f_i, \xi_j, \tau_k))$ is the probability of normal behavior mean to be an outlier of the intrusive data distribution. It is easily seen that $P(f_i|\xi_j, \tau_k) \in [0, 1]$ and $P(f_i|\xi_j, \tau_k) \subset \mathbb{R}$. Moreover, the bigger this probability is, the higher dissimilarity between the two distributions (i.e., normal and attack), and the better the chances of detecting the intrusion ξ_j using the current feature f_i and tuning τ_k.

Whenever a new $P(f_i|\xi_j, \tau_k)$ probability is computed, it is sent to the *1-2 Tier DB* for temporary storage. Once all the possible combinations are exhausted the next Tier is invoked for further processing.

5.2 Statistical Extractor 2^{nd} Tier

The aim of second processing block is to reduce the effect of a particular τ_k tuning from the probability formula that evaluates the chances of a feature to detect an intrusion. As previously mentioned (see Sec. 3) for each of the studied features, multiple tuning factors were employed. This step is not applicable in the case of *Basic Feature* category since no tuning parameters are needed. However, for the other two remaining cases (i.e., *connection based features* and *time based features*) there are 180 tunings for each individual feature that give different probability values. Consequently, it is important to acknowledge the performance of a feature not only against one tuning value but a set of them. This will both make the final probability more reliable, and show how resistant the feature is to different tunings.

Let $P(f_i|\xi_j)$ be the probability of feature f_i to detect attack ξ_j, defined as follows:

$$P(f_i|\xi_j) = \frac{\sum_{\tau_k \in \mathcal{S}_i} P(f_i|\xi_j, \tau_k)}{|\mathcal{S}_i|} \tag{9}$$

where, $\mathcal{S}_i$ is the set of considered tuning combinations for f_i, and $|\mathcal{S}_i|$ is the size of that set (i.e., number of tunings in the set).

While the chosen tuning values cover a fair range of possible tuning combinations, not all of them will facilitate a good detection of attacks. Furthermore, some of them actually inhibit the features to detect certain attacks. Consequently, forcing the set $\mathcal{S}_i$ to include all the possible tuning combinations would not be the best solution since features that perform mediocre in all tunings might be advantaged over features that perform very well on only a reasonable subset of tunings. The viceversa stands too; reporting the performance on only one tuning parameter would probably advantage the features that are unstable to tunings. Furthermore, the extremes best/worse tuning combination should not be considered too, since they might mislead the final result and are not statistically significant.

To solve this issue, we report the performance of the features over their best 50% tuning combinations excluding the best and worse cases. This will advantage both, features that are stable to different tuning factors, as well as features that do not perform well on a small subset of the tunings that we work with.

The 2^{nd} Tier computes all the $P(f_i|\xi_j)$ probabilities and temporary stores them into the *2-3 Tier DB*. Once all the possible combinations are exhausted the final Tier is invoked for further processing.

5.3 Statistical Extractor 3^{rd} Tier

The third and final Tier reports the mean performance of the features based on the defined attack categories.

Let $P(f_i|\zeta_m)$ be the probability of feature f_i to detect attacks that belong to ζ_m category, defined as follows:

$$P(f_i|\zeta_m) = \frac{\sum_{\xi_j \in \zeta_m} P(f_i|\xi_j)}{|\zeta_m|}, \qquad \zeta_m = \{\text{DoS}, \text{Probe}, \text{U2R}, \text{R2L}\} \tag{10}$$

where, $|\zeta_m|$ represents the number of attacks in that category.

This processing will advantage the features that not only accurately detect the attacks from the given ζ_m category, but also the ones that detect the most intrusions that exist in this category. Once the final processing step is done, the features are sorted by their $P(f_i|\zeta_m)$ and reported to the user.

6 Experimental Results

As proof of concept for our statistical evaluation method, we report our findings on the DARPA Intrusion Detection and Evaluation Dataset created by MIT

Lincoln Laboratory [2]. To our knowledge, this dataset is the only public (labeled) dataset in the literature that suits our needs. The dataset has a fair amount of normal and malicious data captured over a period of 5 weeks.

Based on the attack category proposed by MIT, there are 5 main types of attacks in this dataset as follows: Denial of Service (DoS) attacks, Probing attacks, Remote to Local (R2L) attacks, User to Root (U2R) attacks, and Data attacks [7,3,8,9,2]. While the first four types represent different attack mechanisms that an intruder might use to compromise a host, the final attack category (i.e., Data) describes the goal of the attacker and thus no experimental results are presented in this case. Furthermore, the U2R category is invisible to network level features. This assertion is also confirmed by our experiments since none of the features managed to identify any attack. Thus, no experimental results are reported for this category too. Finally, the attacks contained in the R2L category are application level attacks. These types of attacks are not directly visible from the network level unless the payload of the packet is inspected. Consequently, our experimental results are presented only for DoS and Probing attacks.

Most of the attacks that exist in the DARPA 99 dataset are in the last two weeks of the dataset (i.e., Week 4 and Week 5). The traffic was captured using two sniffers and stored into tcpdump files. One of the sniffer was placed inside the network, while the second one was placed at its border. This allowed all traffic inside the network as well as the one exchanged with the outside network to be captured. Thus, there are two tcpdump files for each day except for the second day of the forth week where no inside tcpdump file is provided. Therefore, there are 19 tcpdump files that need to be analyzed for each feature tuning combination. Moreover, as described in Sec. 3, there are 180 tuning combinations for each of the *time based features* and *connection based features* categories. Since the proposed feature extractor (see Sec. 4) can construct in parallel all the network features that are analyzed, there were needed 180 processing steps over each of the input tcpdump files. This lead to a total of 3420 feature extraction jobs that needed to be completed for collecting the final results. These jobs were handled by 30 processors of a Sun V60 supercomputer cluster that managed to exhaust all of them within approximately one month of constant running.

Our experimental results statistically illustrate that our feature evaluation criteria can successfully be applied to analyze the importance of features on the detection process. We statistically show that some of the network features are more suitable than others in the detection process. Moreover, features that tend to detect a particular type of attack may not be useful at all in the detection of other types. Furthermore, the higher the attack is in the TCP/IP architecture model, the smaller is the chance of detecting that attack by monitoring the features extracted at the Transport, Network and Network Access layers. For instance, in general, the computed $P(f_i|\zeta_m)$ probability for DoS and Probe attacks is higher than R2L attacks, while in the case of U2R attacks this probability is close to zero.

In our analysis we use *'Current Connection'* label in the context of a feature whenever that feature is defined based on the connection that the currently

sniffed packet belongs to. Next, *'SrcIP and DstIP Hosts'* label is used whenever a feature refers to all the existing connections between the two IPs that are found in the currently sniffed packet. Thus, the 'SrcIP' and 'DstIP' refer to the source and target IPs of the connection that the currently sniffed packed belongs to. The label *'SrcIP Host'* is used if the current feature analyzes all the connections that exist in the network and have as source or target IP the SrcIP Host. Similarly, the *'DstIP Host'* label is used when the current features monitors all the connections that exist in the network and have as source or target IP the DstIP Host. Finally, the label *'The Network'* is used whenever the current feature is used to describe the wealth of network in general.

Most suitable features for the detection of DoS attacks monitor either all the connections created between two hosts (i.e., *'SrcIP and DstIP Hosts'*), one host and the network (i.e., *'SrcIP Host'*, and *'DstIP Host'*), and the network itself (i.e., *'The Network'*). Furthermore, it seems that the features extracted from ICMP protocol are quite good in detecting DoS and Distributed DoS attacks that uses this protocol(see Table 1). As expected, in the case of Probing attacks, most of the features target connections that were closed by reset, connections that are in a partially open phase (e.g., the 3-way handshaking is incomplete), ICMP packets that report destination unreachable, or echo reply requests to name some. For detecting vertical scanning attacks most suitable features refer to all connections established between two hosts (i.e., *'SrcIP and DstIP hosts'*), whereas for detecting the horizontal ones the most suitable features are those depicting the interaction between a host and the rest of the network (see Table 2).

It is known that different networks may have different optimal tunings for the same feature. Thus, finding and reasoning about the best tuning combination that may be used in a certain network is out of the scope of this current work. However, for future work, this type of study could be done if data is considered from a pool of several different networks, and the reasoning about the best tuning combinations is done relatively to the properties of each individual network (e.g., the *connection window size* may depend on the average number of connections that exist in the network).

We summarize our findings in Tables 3, 4, 5, showing the overall percentage of features and their average effectiveness (i.e., $\overline{P(f_i|\zeta_m)}$) in detecting the main types of attacks. The summary is reported based on the domain abstraction that the features apply to (i.e., *'Current Connection'*, *'SrcIP and DstIP Hosts'*, *'SrcIP Host'*, *'DstIP Host'*, and *'The Network'*).

Due to the diversity of the data and the infinite amount of features that could be extracted from it, the authors recognize that our summary is dependent not only on the selected features, but also on the selected dataset. However, we argue that the set of features that are analyzed cover a fair range of possible combinations and that the experimental results represent just a proof of concept of our evaluation technique. Table 3 presents a general view over all the considered features regardless of their underlying implementation. As seen, for DoS attack, the features extracted from *'SrcIP and DstIP Hosts'* information represent 37% of all the features that detect this category; however, despite the small

Table 1. The top features for the DoS detection sorted by their performance

No	$P(f_i\|\zeta_m)$	Impl. Type	Abstraction	Description
1	0.856	Conn.	SrcIP Host	number of ICMP bytes sent by SrcIP
2	0.853	Conn.	SrcIP Host	number of ICMP packets sent by SrcIP
3	0.832	Conn.	Current Connection	number of ICMP bytes sent by SrcIP to DstIP
4	0.832	Conn.	Current Connection	number of ICMP bytes exchanged
5	0.832	Conn.	SrcIP and DstIP Hosts	number of ICMP bytes sent by SrcIP to DstIP through multiple connections between the two hosts
6	0.829	Conn.	SrcIP and DstIP Hosts	number of ICMP packets sent by SrcIP to DstIP through multiple connections between the two hosts
7	0.828	Conn.	Current Connection	number of ICMP packets sent by SrcIP to DstIP
8	0.828	Conn.	Current Connection	number of ICMP packets exchanged
9	0.759	Conn.	DstIP Host	number of ICMP bytes received
10	0.721	Time	SrcIP and DstIP Hosts	number of TCP packets sent by SrcIP to DstIP with synchronize flag
11	0.711	Time	DstIP Host	number of TCP connections created by any host using any port to connect to DstIP on any port but CSrcPort
12	0.711	Time	DstIP Host	number of TCP connections that use DstIP
13	0.690	Conn.	The Network	number of ICMP packets
14	0.686	Time	SrcIP and DstIP Hosts	number of TCP packets received by SrcIP from DstIP with synchronize flag
15	0.671	Time	SrcIP and DstIP Hosts	number of TCP connections created by SrcIP using any port to connect to DstIP on any port
16	0.665	Time	SrcIP and DstIP Hosts	number of TCP connections between SrcIP and DstIP
17	0.665	Time	DstIP Host	number of TCP packets sent by DstIP with synchronize flag
18	0.659	Time	DstIP Host	number of TCP packets received by DstIP with synchronize flag
19	0.642	Conn.	DstIP Host	number of ICMP packets received by DstIP
20	0.611	Time	SrcIP Host	number of TCP packets sent by SrcIP with synchronize flag
21	0.592	Conn.	DstIP Host	average number of UDP bytes per packet received by DstIP
22	0.588	Conn.	SrcIP and DstIP Hosts	number of TCP connections created by SrcIP using any port to connect to DstIP on any port but CDstPort
23	0.555	Time	SrcIP Host	number of TCP connections created by SrcIP using any port to connect to any other host on any port
24	0.555	Time	SrcIP Host	number of TCP connections created by SrcIP
25	0.555	Time	SrcIP and DstIP Hosts	number of TCP connections created by SrcIP using any port to connect to DstIP on any port but CDstPort
26	0.523	Conn.	SrcIP and DstIP Hosts	number of TCP packets sent by SrcIP to DstIP with synchronize flag
27	0.503	Conn.	SrcIP and DstIP Hosts	number of TCP connections between SrcIP and DstIP
28	0.481	Conn.	SrcIP and DstIP Hosts	average number of TCP bytes per packet sent by SrcIP to DstIP
29	0.476	Conn.	SrcIP Host	average number of TCP bytes per packet sent by SrcIP
30	0.470	Time	DstIP Host	number of TCP connections created by any host using any port to connect to DstIP on the CDstPort

number (i.e., 13%) the most effective features are the ones that characterize the *'Current Connection'* (i.e, $\overline{P(f_i|\text{DOS})} = 0.830$). This is mainly due to the single and multiple connection DoS attacks that generally exhibit a strong behavior at the network level. Furthermore, in the case of Probing attacks, the best solution is to monitor the behavior of each individual host. It is not surprising that the features defined on *'SrcIP Host'* constitute the 47% of all the listed features while also being the most effective ones (i.e.,$\overline{P(f_i|\text{Probe})} = 0.505$).

Table 2. The top features for the Probing detection sorted by their performance

No	$P(f_i\|\zeta_m)$	Impl. Type	Abstraction	Description
1	0.762	Conn.	SrcIP Host	number of TCP connections created by SrcIP which were closed with RST Flag
2	0.755	Time	SrcIP Host	number of ICMP echo reply packets sent by SrcIP
3	0.742	Conn.	SrcIP Host	number of TCP packets received by SrcIP with RST flag
4	0.618	Conn.	SrcIP and DstIP Hosts	number of TCP connections created by SrcIP using any port to connect to DstIP on any port but CDstPort
5	0.583	Time	SrcIP Host	number of TCP connections that use SrcIP which were closed with RST Flag
6	0.582	Time	SrcIP Host	number of TCP packets received by SrcIP with RST flag
7	0.565	Conn.	DstIP Host	number of TCP packets sent by DstIP with RST flag
8	0.535	Time	SrcIP and DstIP Hosts	number of TCP packets received by SrcIP from DstIP with RST flag
9	0.496	Conn.	SrcIP Host	number of ICMP connections created by SrcIP
10	0.491	Time	SrcIP Host	number of ICMP connections created by SrcIP
11	0.484	Time	The Network	the percent of TCP connections that were reset against all existing connections
12	0.456	Conn.	DstIP Host	number of TCP connections created by any host using any port to connect to DstIP on any port but CDstPort
13	0.445	Time	SrcIP Host	average number of ICMP bytes per packet received by SrcIP
14	0.441	Conn.	DstIP Host	number of TCP connections created by DstIP which were closed with RST Flag
15	0.439	Conn.	The Network	number of ICMP connections that have at least one echo reply message
16	0.420	Time	SrcIP Host	number of ICMP packets received by SrcIP
17	0.414	Conn.	SrcIP and DstIP Hosts	number of TCP packets sent by SrcIP to DstIP with reset flag
18	0.411	Time	The Network	number of UDP connections that exist in the network and use different ports than the CDstPort
19	0.409	Conn.	The Network	number of TCP connections the did not finish yet the 3-way handshake
20	0.376	Time	SrcIP Host	number of ICMP bytes received by SrcIP
21	0.371	Conn.	DstIP Host	average number of ICMP bytes per second received by DstIP
22	0.359	Time	SrcIP Host	number of UDP connections created by HostY using any port to connect to any port but CDstPort on any other hosts
23	0.353	Conn.	SrcIP Host	number of TCP packets sent by SrcIP with RST flag
24	0.353	Conn.	SrcIP Host	number of TCP connections created by SrcIP using any port to connect to any other host on any port but CDstPort
25	0.350	Conn.	Current Connection	number of ICMP destination unreachable packets received by SrcIP from DstIP in the current connection
26	0.350	Conn.	SrcIP and DstIP Hosts	number of ICMP destination unreachable packets received by SrcIP from DstIP considering multiple connections
27	0.350	Conn.	SrcIP Host	number of ICMP destination unreachable packets received by SrcIP
28	0.347	Conn.	DstIP Host	number of TCP connections received by DstIP that do not start with SYN Flag
29	0.338	Conn.	The Network	number of TCP connections that connect to CDstPort on various hosts in the network, and did not finish the 3-way handshake
30	0.336	Time	SrcIP and DstIP Hosts	number of TCP connections created by SrcIP using any port to connect to DstIP on any port but CDstPort

Tables 4 and 5 present the same type of information, but separated by the type of implementation used to construct the features. Consequently, Table 4 further presents statistics regarding the *Connection based features*, while Table 5 presents the same statistics based on the *Time based features*. As a general remark, the *Connection based features* tend to be more effective in attack detection than the *Time based features*. This is especially observable in the case of *'Current*

Table 3. Summary of all the studied features regardless of their implementation

Abstraction	DoS		Probing	
	%	$P(f_i\|\text{DoS})$	%	$P(f_i\|\text{Probe})$
Current Connection	13%	**0.830**	3%	0.350
SrcIP and DstIP Hosts	**37%**	0.641	17%	0.451
SrcIP Host	20%	0.651	**47%**	**0.505**
DstIP Host	27%	0.651	17%	0.436
The Network	3%	0.690	17%	0.416

Table 4. Summary of all the studied *Connection based features*

Abstraction	DoS		Probing	
	%	$P(f_i\|\text{DoS})$	%	$P(f_i\|\text{Probe})$
Current Connection	24%	**0.830**	6%	0.350
SrcIP and DstIP Hosts	**35%**	0.626	17%	0.461
SrcIP Host	18%	0.729	**33%**	**0.509**
DstIP Host	18%	0.665	28%	0.436
The Network	6%	0.690	17%	0.395

Table 5. Summary of all the studied *Time based features*

Abstraction	DoS		Probing	
	%	$P(f_i\|\text{DoS})$	%	$P(f_i\|\text{Probe})$
Current Connection	0%	0.000	0%	0.000
SrcIP and DstIP Hosts	**38%**	**0.660**	17%	0.435
SrcIP Host	23%	0.574	**67%**	**0.501**
DstIP Host	**38%**	0.643	0%	0.000
The Network	0%	0.000	17%	0.448

Connection' features for detecting the DoS attacks. In this case, the same set of features manage to outperform all the other ones when implemented using a *connection interval* (i.e., $\overline{P(f_i|\text{DOS})} = 0.830$ in Table 4), whereas, they perform badly when implemented in a *time interval* (i.e, $\overline{P(f_i|\text{DOS})} = 0.000$ in Table 5). Further comparison of the two tables (i.e., Tables 4 and 5), shows that almost all the mean probabilities computed for the *Connection Based features* are higher than the ones computed for the *Time Based features*. Overall, our experimental results illustrate what has always been reported and believed by the research community in the past; however, the novelty of this work is that it proposes a ranking mechanism to evaluate the effectiveness of features against different types of attacks, and that it provides a set of most recommended features to be used for the improvement of detection.

7 Conclusion and Future Work

The feature selection phase is one of the critical tasks that the designers of an IDS / IPS must perform. This phase will later on directly influence the detection performance of the system. The diversity of network features makes almost impossible for a system to create and use all of them. Moreover, there might be features that are insensitive to the attack data, and once feed into the

detection engine might drop the quality of detection. Thus, being able to choose a set of good features for intrusion detection is of great importance.

The aim of this paper is to both provide a statistical technique for studying the importance of a feature in the detection of attacks, as well as to extract a set of recommended features for each of the main types of intrusion attacks (See Appendix 1 and 2 for detail feature listing). We study a set of 673 features that are extracted from the IP, ICMP, TCP and UDP protocols. We restricted our study to those protocols since they are among the most used ones in the Internet. However, our proposed technique can be successfully applied to other application level protocols. For each of the studied features we experimented multiple tuning combinations (i.e., 180) that allowed the final ranking to also consider the sensitivity to different tuning values of the selected features. Therefore, due to the number of features and tunings, as well as the size of the input dataset, we believe that the final results are fairly comprehensive for the protocols that we target.

Our future work will consider other datasets than the one used in this study. However, given the requirements of such dataset (i.e., labels, size, attack diversity, normal traffic) the chances of finding one remain slim (not only for us, but also for the research community). We are also interested in detecting possible interrelations between different features or feature groups in the detection process. Thus, we plan on incorporating this information into our feature evaluation procedure. Another interesting study would be to highlight those features that are not so sensitive to different tunings. Such features would more successfully be used in the detection of attacks since the chances of setting an optimal or near optimal tuning in a given network are slim, the whole process being a trial and error one. Finally, a study will be made on different feature tunings that will aim to propose the most suitable tuning combinations that could be used in the attack detection process.

Acknowledgement

The authors graciously acknowledge the funding from the Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) and through grant RGPN 227441 from the National Science and Engineering Research Council of Canada (NSERC) to Dr. Ghorbani.

References

1. KDD 99. The fifth international conference on knowledge discovery and data mining. Website, Last accessed October (2005), http://kdd.ics.uci.edu
2. DARPA. Darpa intrusion detection and evaluation dataset 1999. Website, Last accessed (February 2006), http://www.ll.mit.edu
3. Das, K.: The development of stealthy attacks to evaluate intrusion detection systems. Master's thesis, MIT Department of Electrical Engineering and Computer Science (June 2000)

4. Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., Tan, P.: Data mining for network intrusion detection. In: Proceedings of NSF Workshop on Next Generation Data Mining, Baltimore, MD, pp. 21–30 (November 2002)
5. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Dokas, P., Kumar, V., Srivastava, J.: Detection of novel network attacks using data mining. In: ICDM Workshop on Data Mining for Computer Security (DMSEC), Melbourne, FL (November, 19) pp. 30–39 (2003)
6. Gibson, H.R.: Elementary statistics. In: Wm. C. Brown Publishers, Dubuque, Iowa (1994)
7. Kendall, K.: A database of computer attacks for the evaluation of intrusion detection syetems. Master's thesis, MIT Department of Electrical Engineering and Computer Science (June 1999)
8. Korba, J.: Windows nt attacks for the evaluation of intrusion detection syetems. Master's thesis, MIT Department of Electrical Engineering and Computer Science, (June 2000)
9. Lippmann, R.P., Cunningham, R.K.: Guide to creating stealthy attacks doe the 1999 darpa off-line intrusion detection evaluation. Project Report IDDE-1, MIT Lincoln Laboratory (June 1999)
10. Zincir-Heywood, A.N., Lichodzijewski, P., Heywood, M.I.: Dynamic intrusion detection using self-organizing maps. In: Proceedings of the 14th Annual Canadian Information Technology Security Symposium - CITSS (May 2002), http://www.sdl.sri.com/projects/emerald/live-traffic.html
11. Powers, A.: Behavior-based ids: Overview and deployment methodology. White Paper Lancope, Inc., 3155 Royal Drive, Building 100, Alpharetta, Georgia 30022, USA (2003), http://www.lancope.com
12. Thomas, J.P., Chebrolu, S., Abraham, A.: Hybrid feature selection for modeling intrusion detection systems. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 1020–1025. Springer, Heidelberg (2004)
13. Sung, A.H., Mukkamala, S.: Identifying important features for intrusion detection using support vector machines and neural networks. In: Proceedings of the IEEE Symposium on Applications and the Internet, January 2003, pp. 209–216. IEEE Computer Society Press, Los Alamitos (2003)
14. Onut, I.-V., Ghorbani, A.A.: A feature classification scheme for network intrusion detection. In: Proceedings of the International Journal of Network Security, vol. 5, pp. 1–15 (July 2007)
15. Stolfo, S.J., Lee, W., Mok, K.W.: Mining in a data-flow environment: Experience in network intrusion detection. In: Proceedings of the 5 International Conference on Knowledge Discovery and Data Mining, pp. 114–124 (1999)
16. Walpole, R.E.: Elementary Statistical Concepts. In: Macmillan (ed.) 2nd edn. (1983)

E-NIPS: An Event-Based Network Intrusion Prediction System

Pradeep Kannadiga[1], Mohammad Zulkernine[1], and Anwar Haque[2]

[1] School of Computing
Queen's University, Kingston
Ontario, Canada K7L 3N6
{pradeep,mzulker}@cs.queensu.ca
[2] Network Planning
Bell Canada, Hamilton
Ontario, Canada L8P 4S6
anwar.haque@bell.ca

Abstract. Intrusion detection systems (IDSs) can detect and respond to various attacks. However, they cannot detect all attacks, and they are not capable of predicting future attacks. In this research, we propose an automatic intrusion prediction system (IPS) called E-NIPS (Event-based Network Intrusion Prediction System) that can not only detect attacks but also predict future probable attacks. We have utilized network penetration scenarios partitioned into multiple phases depending on the sequences they follow during network penetrations. Each of these phases consists of attack classes that are precursors to attack classes of the next phase. An attack class is a set of attacks that have same the objectives, categorized to generalize network penetration scenarios and to reduce the burden on the prediction engine during intrusion alerts correlation and prediction tasks. Future attacks are predicted based on the attack classes detected in an earlier phase of a penetration scenario. Automatic intrusion prediction provides little but very crucial time required for fortifying networks against attacks, warns network administrators about possible attacks, and reduces the damage caused due to attacks. In this paper, we describe the architecture, operation, and implementation of E-NIPS. The prototype implementation is evaluated based on some of the most commonly occurring network penetration scenarios. The experimental results show that the prototype automatically provides useful information about the occurrence of future attack events.

Keywords: Intrusion detection and prediction, attack classes, network security.

1 Introduction

Attacks on computer systems have continued unabated with more sophistication and variants. Some of the attacks are very stealth and harder to detect, and sometimes even when an attack is detected, there is very little time to respond and defend the target system. IDSs are systems that can detect attacks on computer systems and respond to such attacks [1]. However, IDSs cannot sense and respond to the attacks

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 37–52, 2007.

that might probably occur in future. A system that can not only detect attacks but also sense and predict future attacks is called an intrusion prediction system (IPS). IPSs can be more beneficial than IDSs in taking effective measures to secure computer systems by warning security administrators about future attacks. IPSs can be classified into time and event-based intrusion prediction systems [10]. Time-based IPSs predict the seasonality, trends, and cyclic patterns of attacks from historical audit data. Event-based IPSs make prediction based on the event sequences found in multi-step attacks or observed during network penetrations. In this paper, we present an Event-based Network Intrusion Prediction System (E-NIPS) which utilizes the attack sequences occurring in network penetrations to predict future attacks.

Network penetration refers to all the activities that an attacker performs from outside the network to attack a target inside the network [11]. These activities are a set of attacks carried out in a sequence until the objective of the attacker is met. Network penetration scenarios are partitioned into multiple phases depending on the sequence they occur during a network penetration. Each phase is made up of a set of attack classes that could occur in that phase. An attack class is a set of attacks that are variants of the same attack, having a common objective and a set of attributes called attribute vector associated with it for monitoring the likelihood of the occurrence of that attack class in future. For example, an FTP overflow attack class consists of all varieties of FTP overflow exploit attacks that an attack event detection component of the IPS can detect. The attribute vector that can be associated with this attack class consists of the following information: number of connection to the FTP server, version number, number of login failures, anonymous connections, etc. The generalization of a group of attack alerts into an attack class for correlation reduces the computational burden on the prediction engine during the construction of network penetration scenarios. Intrusion prediction is made based on all the precursors to the future attack classes observed in the network penetration scenarios after the attack class correlation.

The proposed E-NIPS architecture contains the databases that store current and past attack events, software and hardware configuration information, and the new attacks reported external to the network. The intrusion prediction engine of the proposed IPS uses the information stored in these databases to perform intrusion prediction. The prediction engine groups and maps attack events to attack classes, generates network penetration scenarios from the correlation of attack classes, calculates the probability of occurrence of future attack classes belonging to the next phase of a network penetration, and also estimates the extent of penetration. The prediction engine is built on the predefined rules that contain the information about the sets of attack classes along with the probability with which they can follow a particular attack class. After future attack classes are predicted, the elements of attribute vectors associated with them are monitored to detect the most likely attacks to follow.

IPSs foresee future attacks and hence provide the crucial time required for fortifying the network against any attack. In this paper, we present the concepts behind intrusion prediction, the architecture of the IPS, and the implementation details of the prototype IPS. The prototype is evaluated using the most commonly occurring network penetration scenarios. Experimental results show that the prototype automatically provides useful information about the occurrence of future attacks.

Paper Organization. Some of the related works are discussed in Section 2. Section 3 elaborates the phases of a network penetration and discusses the heuristics used in this work. Section 4 presents the architecture of E-NIPS by describing its components. Section 5 describes the implementation details of the prediction engine component of E-NIPS. Section 6 provides the experimental evaluation results of the prototype IPS. Section 7 summarizes the current work and its limitations and outlines the future work.

2 Related Work

Mathematical forecasting techniques are used for various applications like weather and business forecasting. In [2], various forecasting techniques that can be applied for business applications such as profit/loss prediction is discussed. In [9], fuzzy logic is applied for weather forecasting. In [3, 4, 5, 6, 7, 8], various forecasting techniques such as decision tree, Hotelling's T2 test, chi-square multivariate test, markov chain, exponentially weighted moving average, neural networks, software agents, and local linear models have been used for intrusion detection. These techniques predict next user/system behavior based on the available information learnt by the forecasting algorithms during training phase. Any deviation from the behavior predicted by the forecasting algorithms is considered as abnormal, and an alert is generated to inform about the attack. However, the prediction of future intrusive events has not been dealt so far in the works mentioned above.

In [12], the predictive patterns that are rules generated inductively from the event sequences present in audit data are used for intrusion detection. Suppose *P1* and *P2* are events that have already occurred, *F1* and *F2* are events that might occur with some probability x and y, then the rule $\{P1\text{-}P2 \rightarrow (F1=x\%,\ F2=y\%)\}$ is used. The rules represent normal user behavior and any deviation from the normal user behavior is detected by the rules as an attack. Therefore, it is basically an anomaly-based intrusion detection system [1]. The rules used in our work contain information of attack event sequences observed in network penetration scenarios instead of patterns seen in audit data. Furthermore, the rules in our work are used for the correlation of detected attack classes to construct ongoing network penetration scenarios, and the scenarios are also utilized for future attack events prediction instead of the mere task of detection [12].

In [13,14,15,16,17], a graph-based approach is used to relate various attack events or alerts from an IDS with vulnerability and network topology information to construct a multi-stage attack scenario for recognizing attacker intensions. The nature of work presented in [15, 16, 17] are more closely related to our work, and hence, they are discussed in the following paragraphs.

In [15], a work on alert correlation and multi-stage attack scenario construction is proposed. The cluster of alerts that correspond to the same occurrence of an attack are merged into one alert to reduce the number of alerts that are sent to a security administrator and for the ease of alert correlation. During correlation, attack alerts that are related to a multi-stage attack scenario are correlated using rules to generate a complete attack scenario. In our work, we have classified the alerts that are variants of the same attack as an attack class and the classification is also based on the unique

attributes associated with the alerts. This classification of attacks simplifies the domain knowledge about the relationship/dependency between different attacks. The classification process involves merging alerts as discussed in [14] with an additional step of mapping to an attack class. However, attribute vectors and some heuristics used in our work help in making finer-grained prediction of attacks from the predicted attack classes. The use of network penetration scenarios in our work makes it more useful tool for penetration testing and risk assessment.

In [16], a technique for alert correlation and future alert prediction is proposed. Alerts from individual attack steps are correlated to detect multi-step attacks. For the purpose of alert correlation and prediction, attack graphs are built based on interdependencies between attack exploits and security conditions that are required for the corresponding exploit to be executed. Prediction is made based on the new security conditions that would be satisfied from any newly generated alerts. In our work, we have considered network penetration scenario graphs that show all the possible attack classes that follow its precursors. The number of attack classes used in our work during attack class correlation is considerably lower than the number of alerts used in [16] during alert correlation. As a result, attack class correlation is easier to perform. The simplicity of a network penetration scenario graph resulting from attack classes permits the usage of simple predefined rules for network penetration scenario construction and prediction. In our work, by utilizing penetration scenarios partitioned into phases, we can also determine the extent of a network penetration.

In [17], an automatic attack graph construction mechanism and an alert correlation technique using multi-layer perceptron (MLP) and support vector machines (SVM) are presented. The alerts having similar features such as same source and destination addresses are used to compute the correlation probability between alerts. The correlated alerts are then represented as a hyper-alert graph to detect a multi-stage attack. The generated graph combined with vulnerability and topology information is used for target recognition and risk assessment. In our work, attack classes instead of each of the alerts belonging to the different phases of a network penetration are correlated based on predefined rules, and the conditional probabilities of future attacks are predicted. A set of attribute vectors indicates the security conditions related to the predicted attack classes that have to be monitored.

3 Network Penetration

In this section, we describe the network penetration phases using an example penetration scenario and discuss some of the heuristics used for prediction.

3.1 Network Penetration Phases

In this research, we have considered network penetration scenarios [11] that constitute the attacks that an attacker employs to find the way from outside to the target host present inside the network. Some of the important phases of a network penetration and the attacks possible in those phases are discussed below.

Reconnaissance (recon). This is the first phase in a network penetration where an attacker tries to gather information such as network topology, open ports, shared folders, user accounts, and software/hardware configuration information. DNS zone transfer, NSLookup, port/network scanning, operating system and web fingerprinting, and vulnerability scan are some the attacks used by an attacker in this phase.

Exploitation. This phase consists of attacks against the target that exploit the information already gathered in the reconnaissance phase to perform attacks such as buffer overflow, unauthorized access, file manipulation, etc.

Denial of Service (DoS). DoS attacks are the class of attacks that follow after a recon or an exploitation phase. DoS attacks cause disruptions in the services available to users, and hence, these attacks have to be detected as early as possible.

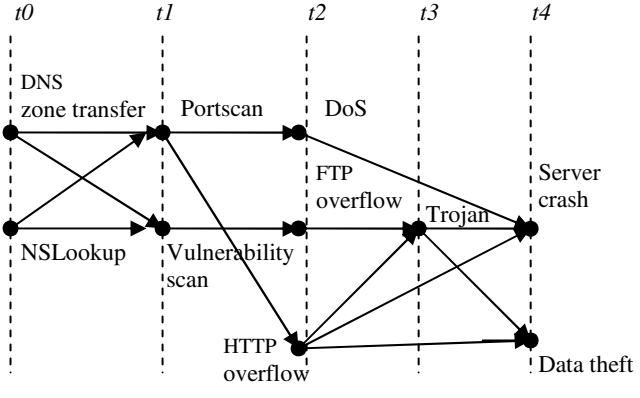


Fig. 1. An example network penetration scenario

Malicious Code. In this phase, attacks such as uploading of distributed DoS agents, Trojan, rootkits, backdoors, virus, or worms could take place.

For intrusion prediction, we identify attack classes within each phases discussed above based on the common objective that attack events are trying to attain. For example [18], all FTP overflow exploits like FTP CMD overflow attempt, FTP PASS overflow attempt, FTP USER overflow attempt, etc. are considered as one class named FTP overflow, and this attack class falls under the exploitation phase of a network penetration. Section 5.1 contains detailed information on attack classification.

An example network penetration scenario is shown in Fig. 1. In this figure, we can see recon attack classes (DNS zone transfer and NSLookup) at time *t0* and (Portscan and Vulnerability scan) at time *t1*, denial of service (DoS) and exploitation attack classes (FTP overflow and HTTP overflow) at time *t2*, malicious code attack class (Trojan) at time *t3*, and the final objective of the attacker (Server crash and Data theft) at time *t4*. The arrows in Fig. 1 indicate the causal relationships between attacks. Based on the attacks that have occurred at time *t0* and *t1*, we can predict attacks possibly occurring at time *t2*. Similarly, based on the attacks detected at time *t0, t1,* and *t2,* we can predict attacks that might occur at time *t3* and *t4*.

3.2 Heuristics for Intrusion Prediction

Several heuristics are used in deciding about the attack class which has the higher probability of occurring in near future [11, 13]. Some of the heuristics used in this work are mentioned below:

1) Port scanning is considered precursor to DoS and exploitation attacks.
2) Vulnerability scanning is considered precursor to attacks on vulnerable software and hardware.
3) Vulnerable software/hardware versions are more prone to attacks. For example, exploits on some of the versions of wu-ftp server/ISS web server already exist and can be easily used against these servers.
4) If any software is recently reported to be vulnerable, then the chances of attacks on this software are higher.
5) Recon activities against specific servers lead to attacks specific to that server. For example, FTP server related recon activities lead to FTP server related attacks like FTP overflow exploits. Similarly, recon activities on a particular system/operating system lead to attacks on that particular system/operating system.
6) A newly reported worm/virus that has infected an external network has higher probability of attacking the network.

Along with the general heuristics listed above other attack specific precursors are used to decide about the probability of a particular attack class occurring in future. For example, suspicious logins or login failures can lead to user account compromise and act as a precursor to attacks such as uploading of rootkits and system file corruption.

4 E-NIPS Architecture

Fig. 2 shows the architecture of E-NIPS. The main components are Host Activity Collector and Reporter (HACR), Network Activity Collector and Reporter (NACR), Internal Activity Database (IAD), Vulnerability Profile Database (VPD), External Activity Database (EAD), and Intrusion Prediction Engine (IPE). Software/hardware configuration and web postings/mailing lists components shown in dotted boxes indicate the information stored in the VPD and the EAD respectively. The HACR and NACR are attack events monitoring components and the IAD, the VPD, and the EAD are the storage components that store attack events, software/hardware configuration, and web postings/ mailing lists respectively. The IPE is the principal intrusion prediction component.

Host Activity Collector and Reporter (HACR). It is a probe placed on every host (1 to *n*) within the network. It is used for monitoring the host for attack events such as system files corruption, presence of sniffers and vulnerable applications, and installation of rootkits/backdoors on the host. The HACRs send any detected attack event to the IAD component for storage.

Network Activity Collector and Reporter (NACR). It is also a probe that can monitor network traffic for malicious attack events such as port/network scanning, DoS attacks, virus or any overflow exploit content in network traffic, communication between Trojan client and server, and suspicious logins. The NACR monitors traffic

to servers such as ftp, http, rpc, snmp, smtp, and pop. It also sends any detected attack event to the IAD component for storage. The attack events detected in the network by the NACR and the attack events detected by the HACRs on every host within the network result in multipoint detection and provide more comprehensive attack detection capability to the IPS. Snort is a widely used open source network intrusion detection tool [18] that has been used as the suspicious network activity detector by the NACR of E-NIPS. Snort is a rule-based intrusion detection system that contains more than 1,500 attack signatures for matching against sniffed network traffic to detect any malicious content in the traffic. Snort generates an alert on the detection of any attack traffic in the network.

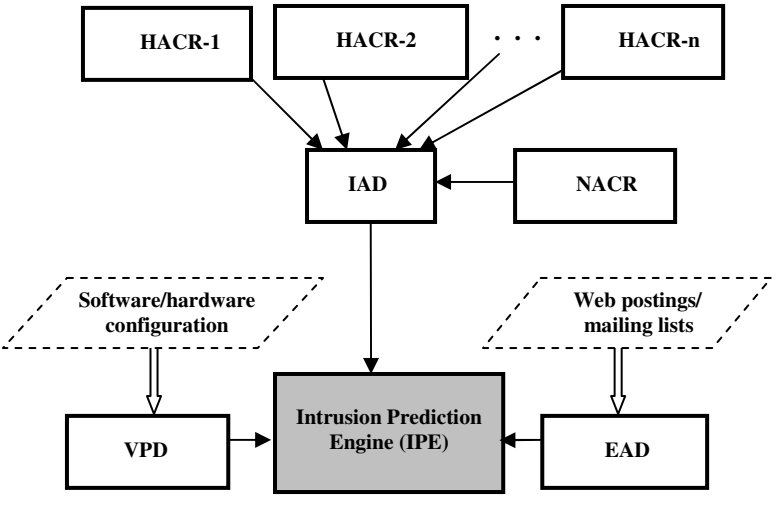


Fig. 2. E-NIPS architecture

Internal Activity Database (IAD). This database stores all the attack events detected in the hosts and the network by the HACRs and the NACR monitoring components. This database contains attack events that have occurred in the past to be used by the IPE to construct network penetration scenarios and predict future attacks.

Vulnerability Profile Database (VPD). This database stores hardware and software configuration information of the internal network environment. The VPD is necessary to apply the heuristics discussed in Section 3.2 and is populated with data from vulnerability scanner tools. The VPD also stores the information of open ports, available services, and operating systems of every monitored host.

External Activity Database (EAD). This database stores the data related to the newly reported attacks collected from external sources like web posting and mailing lists. The EAD data is utilized to defend against those attacks.

Intrusion Prediction Engine (IPE). The IPE is the main prediction component of E-NIPS, and it is discussed elaborately in the next section.

5 Intrusion Prediction Engine

The IPE predicts intrusions based on the events stored in the IAD and the data stored in the VPD and the EAD. The IPE consists of a classifier and a predictor module. The classifier maps detected attack event alerts to attack classes and sends the mapped list of attack classes to the IPE. The predictor module generates network penetration scenarios by correlating attack classes received by the classifier and determines the probability of occurrence of attack classes of future penetration phases. It also estimates the extent of a penetration. The two most important functional modules of the IPE are discussed in the following subsections.

5.1 Classifier

The function of the classifier module is to take the set of attack events from the IAD and determine the attack classes of those events. The successive occurrences of attack events belonging to the same attack class is considered as single occurrence to reduce the number of attack classes sent to the predictor module. The output from the classifier is a set of attack classes that are arranged according to the sequence they have occurred and found in the IAD.

The number of attack events detected by the NACR and the HACRs and the number of potential penetration scenarios considering all the attack events present in the IAD are very high. As a result, it is difficult to manually generate the rules for prediction in the IPE. Therefore, we have grouped a set of attack events into attack classes based on the attack objective and commonality between those attack events. In Fig. 3, we can see different attack classes used by E-NIPS and the sequence they could occur during a network penetration. Fig. 3 shows four separate columns with each column representing different phases of a network penetration with possible attacks in the corresponding phase. For example, all the FTP server related attack events are grouped into four attack classes: FTP overflow attempts, specific version, suspicious login, and DoS attempts as shown in the second column under FTP attacks. All different kinds of FTP overflow exploit attack events detected by the HACRs and the NACR are classified as “FTP overflow” attack class. The attacks against only specific versions of FTP servers (wu-ftp, serv-u) are grouped under a separate attack class, and similarly, all kinds of suspicious login attempts (brute force login attempts, anonymous/guest user login) are grouped as a separate attack class.

The first column of Fig. 3 shows attack classes belonging to initial recon phase of a network penetration. Port scanning, network topology detection, and specific server recon activities occur in this phase. The recon phase is followed by attacks against specific servers such as DNS, FTP, RPC, and HTTP. The attack classes belonging to different servers are shown in the second column. In the third column, different attack classes (Backdoors/DDoS, DoS, and Web application attacks) that could follow attack classes of the second column are shown. The final column of Fig. 3 shows the attack classes that could be considered as final attack objective of the attacker. Data theft or alteration, system malfunctions, denial of service, web pages alteration, configuration change, and account compromise are some of the attack classes that are possible in the final phase of a network penetration.

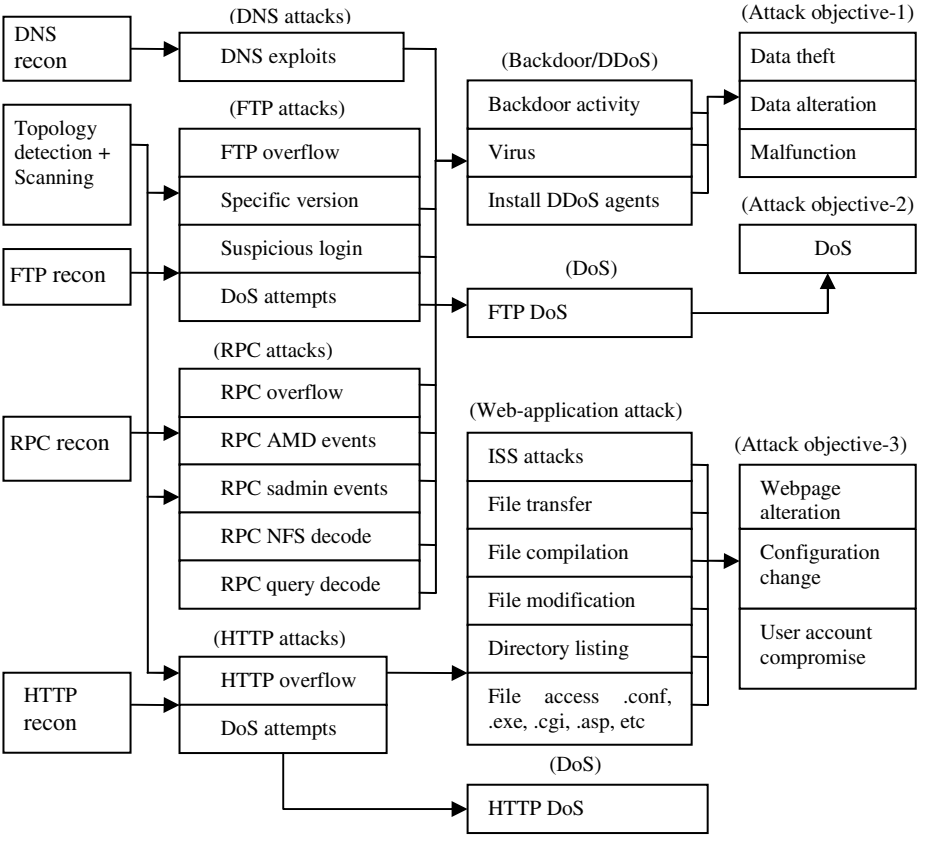


Fig. 3. Attack classes and their sequences during a network penetration as used by the IPE

5.2 Predictor

The predictor correlates the set of attack classes received from the classifier module to construct network penetration scenarios for the purpose of prediction. The predictor module consists of predefined rules to predict future attacks based on the attack classes received from the classifier. Each rule stores information about the set of attack classes and the conditional probability with which they can follow a particular attack class. The rules also identify the attack with the highest probability of occurrence using some of the heuristics discussed in Section 3.2. After a set of attack classes are predicted by the rules, the attribute vectors associated with the predicted attack classes are monitored to identify the most likely attack to follow. For example, after occurrence of attack class *A1,* in Fig. 4, the elements of the attribute vectors of *B1, B2, B3*, *B4,* and *B5* are monitored by the HACR and the NACR. If an activity is detected in the elements associated with attribute vector of attack class *B1,* then the probability of *B1* occurring increases. Finally, the predictor also determines the extent of a network penetration to indicate the attack initiation, attack progress, or attack objective stage of a penetration. The attack classes detected in the recon phase

indicate attack initiation stage and the attack classes that lead to system compromises like those shown in the second and the third column of Fig. 3 indicate attack progress stage. The attack classes that indicate attack objectives as shown in the fourth column fall into the attack objective stage of a network penetration.

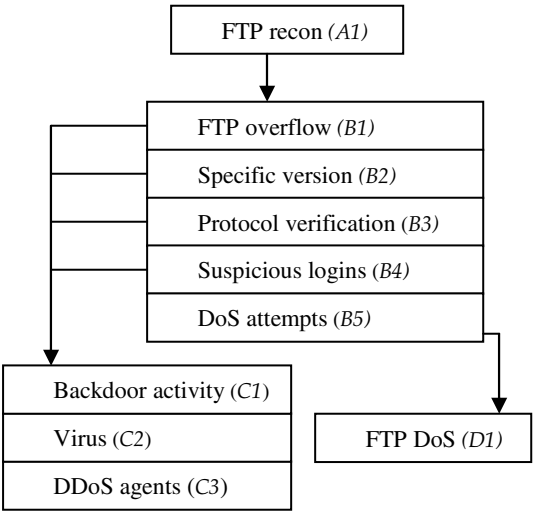


Fig. 4. A penetration scenario used by the IPE

Fig. 4 shows one of the penetration scenarios taken from Fig. 3 that can be visualized as a graph with each attack class as a separate node and the arrows indicating the causal relationships between attack classes. In Fig. 4, we can see a FTP recon activity *(A1)* is a precursor to FTP server related attack classes *(B1, B2, B3, B4, B5)* which in turn leads to backdoor /virus/DDoS *(C1, C2, C3)* and FTP DoS *(D1)* attack classes. In this scenario, we consider *B1, B2, B3, B4,* and *B5* are mutually exclusive and they follow attack class *A1* with equal probability. Therefore, the probability of *B1* occurring after *A1* denoted as **P**(*B1*) is equal to $1/5$, and similarly, **P**(*B2*), **P**(*B3*), **P**(*B4*), and **P**(*B5*) are all equal to $1/5$. *C1, C2,* and *C3* can follow events *B1, B2, B3,* and *B4* with equal probability. Therefore, the conditional probabilities of attack class *C1* following *B1, B2, B3,* and *B4* are shown by Equation 1.

$$\mathbf{P}(C1/B1) = \mathbf{P}(C1/B2) = \mathbf{P}(C1/B3) = \mathbf{P}(C1/B4) = 1/3 \quad (1)$$

The conditional probabilities such as **P**(*C2*/*B1*) and **P**(*C3*/*B1*) can also be calculated similarly. *D1* alone follows *B5* and hence **P**(*D1*/*B5*) equals 1. Equation 2 is used to calculate the probability of occurrence of attack class *C1* starting from *A1*. Similarly, probabilities **P**(*C2*) and **P**(*C3*) are also calculated.

$$\mathbf{P}(C1) = \mathbf{P}(C1/B1)\mathbf{P}(B1) + \mathbf{P}(C1/B2)\mathbf{P}(B2) + \mathbf{P}(C1/B3)\mathbf{P}(B3) + \mathbf{P}(C1/B4)\mathrm{P}(B4) \quad (2)$$

In many cases, subjective probabilities could be used based on the data from the VPD, the EPD, and the heuristics discussed in Section 3.2. The set of future attack

classes that follow an attack class from an earlier phase along with their probabilities are programmed into the IPE using a set of rules of the form shown in Fig. 5. These rules are derived from the scenario shown in Fig. 4.

Rule 1: *A1* → *B1, B2, B3, B4, B5 (x1)*
Rule 2: *B1* | *B2* | *B3* | *B4* → *C1, C2, C3 (x2)*
Rule 3: *B5* → *D1 (x3)*

Fig. 5. Sample rules used in the IPE

Rule 1 of Fig. 5 shows that if attack class *A1* has occurred then *B1*, *B2*, *B3*, *B4* or *B5* could occur with probability *x1*. The value of *x1* programmed in Rule 1 is 1/5 if *B1*, *B2*, *B3*, *B4* and *B5* are considered to occur with equal probability or a value based on subjective probability could be assigned to *x1*. Rule 2 shows that if *B1*, *B2*, *B3*, or *B4* occur then *C1*, *C2*, or *C3* could follow with probability *x2*. The value of *x2* programmed in Rule 2 is 1/3 if *C1*, *C2*, and *C3* are considered to occur with equal probability or a value based on subjective probability could be assigned to *x2*. Similarly, Rule 3 of Fig. 5 shows that when *B5* occurs then *D1* is the most likely event to occur, and hence, *x3* is *1* or based on subjective probability.

Applying the types of rules shown in Fig. 5, the probability calculator generates penetration scenarios from the attack class list received from the classifier module. Suppose (*A1*, *B1*) is the attack class list received from the classifier module. Using Rule 1 and Rule 2 of Fig. 5 the penetration scenario Scenario-A of Fig. 6 is generated. The probability value *x4* in Scenario-A of Fig. 6 is calculated using Equation 2 considering the occurrence of both *A1* and *B1*. If there exists an additional rule such as Rule 4: *C1* → *D1, D2, D3(x5)* and (*A1*, *B1*, *C1*) class list is received from the classifier module then using Rule 1, Rule 2, and Rule 4 Scenario-B of Fig. 6 would be generated. The probability of occurrence of *A1*, *B1*, and *C1* is considered in calculating *x6* of Scenario-B.

Scenario-A: *A1* → *B1* → *C1, C2, C3 (x4)*
Scenario-B: *A1* → *B1* → *C1* → *D1, D2, D3 (x6)*

Fig. 6. Example scenarios generated by the IPE

The scenarios shown in Fig. 6 are generated from the rules (see Fig. 5) programmed into the IPE. However, scenarios can also be generated using similar features such as attack origin address, target address, and port numbers found in the attack events. These kinds of scenarios show the attacks originating from one source to a single or multiple destinations.

6 Experimental Evaluation

The prototype E-NIPS is evaluated using various commonly occurring network penetration scenarios. However, we explain the evaluation process in detail using only two penetration scenarios (see Appendix A). Scenario-I is a penetration scenario

consisting of three phases simulated using attack tools [11, 20]. These attack tools generate attack events belonging to different phases of the penetration scenario. The experimental setup for this scenario consists of a network of hosts providing different services and the configuration information of the network is stored in the VPD. Scenario-II is a penetration scenario consisting of five penetration phases based on a scenario observed in 1999 MIT intrusion detection evaluation project [19]. We are aware of the limitations of this data source [21]. However, we just have chosen the most commonly occurring scenario and the general limitations of this dataset do not affect our evaluation objective. The alerts generated by the HACR and the NACR of E-NIPS during simulation of these penetration scenarios are stored in the IAD and used for the purpose of intrusion prediction. We have observed successful performances of E-NIPS in both the scenarios.

From the experimental evaluation, we have observed that the implemented IPS provides ample information about attack classes that could occur in future. Moreover, in some cases, with the help of the data from the VPD, the EPD, and the heuristics discussed in Section 3.2, it can also predict highly probable attack events within the attack class. The creation of attack classes from attack alerts have made the correlation and prediction tasks simpler. In the experiments, the ongoing penetration phase was detected and prediction was made almost instantaneously. This shows that the IPE provides a time window of atleast few minutes for an administrator to defend the network. However, in real-world scenarios, the time gap between the time at which E-NIPS makes prediction about a future attack event and the time when that future attack event is actually detected can vary from few seconds to many hours or days. An attacker can launch the next set of attack events according to his/her own convenience, and hence, predicting the time of forthcoming attack events is difficult. However, based on the timings observed in the attack events from the earlier phases of a network penetration, the time of future attacks events may be predicted using various forecasting algorithms [2].

7 Conclusions and Future Work

This paper presents the concepts, method, and architecture of an intrusion prediction system along with the implementation and evaluation of a prototype IPS based on various simulated network penetration scenarios. The attack events having a common objective are grouped into one attack class, and instead of every attack event, we have considered attack classes during intrusion prediction. The sequence in which the attack classes occur during a network penetration and the conditional probability with which they occur is programmed into the prediction engine of the IPS as a set of predefined rules. Intrusion prediction is made considering various parameters related to the precursors to future attacks, vulnerability profile, reports of external worm/virus activities, and a set of heuristics. The IPS consists of host and network monitors that provide multi-point detection of attack events.

The evaluation results show that this IPS can predict future attack events with reasonable accuracy and also estimate the extent of a penetration. Attack classes are used to reduce the number of penetration scenarios that have to be programmed into the prediction engine of the IPS. Attack class correlation and intrusion prediction provides

information of a network penetration and helps to defend the network. This tool can help in testing network penetrations and addressing the security risks of a network.

E-NIPS can construct and predict only the network penetration scenarios programmed as predefined rules in the IPE. Without sufficient knowledge of existing network penetrations, it is difficult to program these rules. The proposed IPS would not be very useful if there is not enough time gap between network penetration phases like in zero day attack scenarios. In future, we intend to investigate the advantages of using various prediction models such as neural networks, decision tress, and markov chains in the prediction engine. We will also develop time-based prediction engine for forecasting seasonality, trends, and cycles in intrusion data. Predicting the time of occurrence of future attack events will also be considered in the time-based prediction engine.

Acknowledgments

The authors would like to thank Dr. Lingui Wang of Concordia University, Canada for his comments on an initial version of this work. This research work is partially funded by Bell Canada through Bell University Laboratories (BUL) and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

1. Mukherjee, B., Heberlein, L.T., Levitt, K.N.: Network intrusion detection. IEEE Network 8, 26–41 (1994)
2. Arsham, H.: Time-critical decision making for business administration. University of Baltimore, Maryland, USA (Accessed January 2006), http://home.ubalt.edu/ntsbarsh/Business-stat/stat-data/Forecast.htm
3. Ye, N., Li, X., Chen, Q., Emran, S.M., Xu, M.: Probabilistic techniques for intrusion detection based on computer audit data. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 31(4) (2001)
4. Ye, N., Chen, Q., Borror, C.M.: EWMA forecast of normal system activity computer intrusion detection. IEEE Transaction on Reliability 53(4) (December 2004)
5. Ramasubramanian, P., Kannan, A.: Quickprop neural network ensemble forecasting framework for a database intrusion prediction system. Neural Information Processing - Letters and Reviews 5(1) (2004)
6. Pikoulas, J., Buchanan, W.J., Mannion, M., Triantafyllopoulos, K.: An agent-based bayesian forecasting model for enhanced network security. In: Proc. of the Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems-ECBS, pp. 247–254. Los Alamitos, CA, USA (April 2001)
7. Govindu, S.K.: An intelligent mobile agent-based intrusion forecasting system. March 2005 (Accessed January 2006), http://www.securitydocs.com/library/3110
8. Hu, P., Heywood, M.I.: Predicting intrusions with local linear models. In: Proc. of the IEEE International Joint Conference on Neural Networks, vol. 3, pp. 1780–1785 (2003)
9. Maner, W., Joyce, S.: WXSYS Weather Lore + Fuzzy Logic = Weather Forecasts. Presented at the 1997 CLIPS Virtual Conference, 1997 (Accessed January 2006), http://web.cs.bgsu.edu/maner/wxsys/wxsys.htm
10. Zulkernine, M., Haque, A., Desroches, M.: Will I be attacked - forecasting network intrusions. In: The 16th IEEE International Symposium on Software Reliability Engineering (ISSRE), Chicago, Illinois, USA, vol. 4, pp. 9–10 (November 2005)

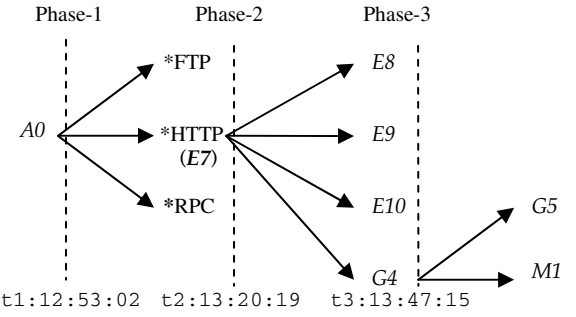
11. McClure, S., Scambray, J., Kurtz, G.: Hacking Exposed, 5th edn. Mc-Graw Hill, U.S.A (2005)
12. Teng, H.S., Chen, K., Lu, S.C.: Security audit trail analysis using inductively generated predictive rules. In: Proc. of the Sixth Conference on Artificial Intelligence Applications, New Jersey, pp. 24–29 (March 1990)
13. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: Proc. of the 1998 Workshop on New Security Paradigms, pp. 71–79. Virginia, U.S.A (1998)
14. Levitt, K., Templeton, S.J.: A requires/provides model for computer attacks. In: Proc. of the 2000 Workshop on New Security Paradigms, Cork, Ireland (February 2001)
15. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection framework. In: Proc. of 2002 IEEE Symposium on Security and Privacy, pp. 202–215. Oakland, California, USA (2002)
16. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlating, hypothesizing, and predicting network intrusion alerts. Computer Communications 29(15), 2917–2933 (2006)
17. Zhu, B., Ghorbani, A.: Alert correlation for extracting attack strategies. International Journal of Network Security 3(3), 244–258 (2006)
18. Roesch, M.: Snort – lightweight intrusion detection for networks. In: Proc. of USENIX LISA 99, Seattle, Washington, USA (1999)
19. Lincoln, M.I.T.: Laboratory, 2000 DARPA intrusion detection scenario specific datasets (Accessed January 2006), http://www.ll.mit.edu/IST/ideval/data/2000/2000_data_index.html
20. Security Forest :ToolTree (Accessed August 2006), http://www.securityforest.com/wiki/index.php/Category
21. McHugh, J.: Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Off-line Intrusion Detection System Evaluation as Performed by Lincoln Laboratory. ACM Transactions on Information and System Security 3(4), 262–294 (2000)

Appendix: A

A.1 Scenario-1

The output generated by the predictor module of the IPE after each phase of Scenario-I is shown in Fig. 7. The time `t1`, `t2`, and `t3` represented in `HH:MM:SS` format in Fig. 7 indicate the start of each penetration phase. The generated alerts are mapped into four attack classes *A0, E7, G4,* and *G5* by the classifier module.

In Phase-1, the attacker performs network scanning and determines the presence of web server indicated by open port 80. RPC attacks, FTP attacks, and HTTP attacks are predicted by the IPE to occur in the next phase since the VPD is programmed for this experiment to indicate only these services running on the network. In Phase-2, indicated by WEB file access (*E7*) attack class, a potential vulnerable application file is accessed by the attacker during a buffer-overflow exploit attack on web server. The attack classes following *E7* and related to web server attacks are predicted. In Phase-3, indicated by ATK-RSP directory listing attempt *(G4*) attack class, directory listing attempt is detected in the response traffic indicating the compromise of the web server from the attack from Phase-2. Among the attack classes predicted after Phase-3, ATK-RSP command-execution attempt (*G5*) is found to be executed by the attacker.



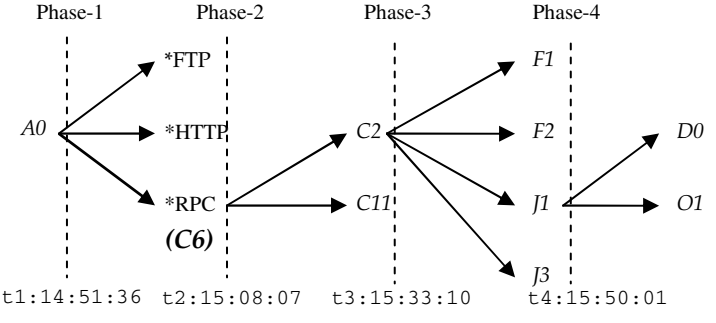
A0 - Scanning, *FTP – all FTP attacks, *HTTP – all HTTP attacks, *RPC – all RPC attacks, *E7* - WEB file access, *E8* - WEB remote-connect attempt, *E9* - WEB dos attempt, *E10* - cross-scripting, *G4* - ATK-RSP directory listing attempt, *G5* - ATK-RSP command-execution attempt, *M1* - DoS attempt.

Fig. 7. Output from the IPE for Scenario-I after each phase of penetration

A.2 Scenario-II

The output generated by the predictor module of the IPE after each phase of Scenario-II is shown in Fig. 8. Here, Phase-1 and Phase-2 fall in the attack initiation stage, Phase-3 and Phase-4 are called attack progress stage, and Phase-5 is considered attack objective stage of a network penetration. The alerts were mapped into five attack classes *A0, C6, C2, J1,* and *D0* by the classifier module of the IPE.

In Phase-1, the attacker performs network scanning using ICMP echo-requests and from the received ICMP echo-replies determines the hosts and services that are



A0 - Scanning, *FTP – all FTP attacks, *HTTP – all HTTP attacks, *RPC – all RPC attacks, C6 - RPC sadmin query decode, *C2* - RPC sadmin overflow, *C11* - RPC exploits, *F1* - Backdoor activity, *F2* - Virus, *J1* - RSH exploit, *J3* - REXEC overflow, *D0* - DoS attempt, *O1* - Data theft.

Fig. 8. Output from the IPE for Scenario-II after each phase of penetration

running within the network. RPC attacks, FTP attacks, and HTTP attacks are predicted by the IPE since the VPD is programmed for this experiment to indicate only these services running on the network. In Phase-2, indicated by RPC sadmin query decode (*C6*) attack class, the hosts discovered in Phase-1 are probed to detect if those hosts are running "sadmin" administration tool. The intention of the attacker being to use sadmin exploits against those hosts, and hence, RPC sadmin overflow *(C3)* and other RPC exploits *(C11*) attack classes are predicted.

In Phase-3, indicated by RPC sadmin overflow *(C2)*, the attacker attempts sadmin remote buffer overflow exploit on those hosts with sadmin tool and creates new root user accounts on the hosts on which the exploits are successful. The system is compromised at this stage, and therefore, Backdoor activity *(F1)* and Virus *(F2)* attack classes, remote connection attack classes RSH exploit *(J1)* and REXEC overflow *(J3*), and DoS attempt (*D0*) attack classes are predicted to follow. In Phase-4, indicated by RSH exploit (*J1*), the attacker telnets to the compromised hosts and uploads DDoS agent on the systems that are compromised in the previous phase. DoS attempt (*D0*) and Data theft (*O1*) attack classes are predicted to follow this phase. In Phase-5, indicated by DoS attempt (*D0*), the attacker launches DDoS attacks on victim host using DDoS agents installed in the previous phase.

Enabling Fairer Digital Rights Management with Trusted Computing*

Ahmad-Reza Sadeghi[1], Marko Wolf[1], Christian Stüble[2], N. Asokan[3], and Jan-Erik Ekberg[3]

[1] Horst-Görtz-Institute for IT-Security, Ruhr-University Bochum, Germany
{sadeghi,mwolf}@crypto.rub.de
[2] Sirrix AG security technologies, Germany
stueble@sirrix.com
[3] Nokia Research Center, Helsinki, Finland
{n.asokan,jan-erik.ekberg}@nokia.com

Abstract. Today, digital content is routinely distributed over the Internet, and consumed in devices based on open platforms. However, on open platforms users can run exploits, reconfigure the underlying operating system or simply mount replay attacks since the state of any (persistent) storage can easily be reset to some prior state. Faced with this difficulty, existing approaches to Digital Rights Management (DRM) are mainly based on preventing the copying of protected content thus protecting the needs of content providers. These inflexible mechanisms are not tenable in the long term since their restrictiveness prevents reasonable usage scenarios, and even honest users may be tempted to circumvent DRM systems.

In this paper we present a security architecture and the corresponding reference implementation that enables the secure usage and transfer of stateful licenses (and content) on a virtualized open platform. Our architecture allows for openness while protecting security objectives of both users (flexibility, fairer usage, and privacy) and content providers (license enforcement). In particular, it prevents replay attacks that is fundamental for secure management and distribution of stateful licenses. Our main objective is to show the feasibility of secure and fairer distribution and sharing of content and rights among different devices. Our implementation combines virtualization technology, a small security kernel, trusted computing functionality, and a legacy operating system (currently Linux).

Keywords: Trusted Computing, security architectures, stateful licenses.

1 Motivation

Timo was about to board a train home when he noticed an advertisement for a wireless kiosk selling the first album from a new band. He took out his music

* Full version appears as a technical report HGI-TR-2007-002 in [24].

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 53–70, 2007.

phone, connected the kiosk which was already visible in his music gallery application, and with a few clicks downloaded a preview copy of the lead song in the album. While on board the train, Timo listened to the song and liked it so much that he listened to it once more. When he tried to listen a third time, the phone told him that he had finished the free previews, but can buy a full license. He bought the full license with a few more clicks and could listen to the song with no constraints. When he got home, he transferred the song to his home stereo system. When Anna visited Timo, he played the new song to her. She wanted a copy of her own. Timo used the remote control of his stereo system to lend a copy to Anna's music phone for a week. Timo's copy of the song remained disabled for a week while Anna was enjoying the song.

This and other similar scenarios for trading and using digital goods involve policies whose enforcement requires the enforcement mechanism to securely maintain state information about past usage or environmental factors. They can be enforced by using *stateful licenses*. Some e-business applications already deploy such (mostly proprietary) stateful licences to sell certain digital goods (online video, music tracks, software), for limited use (number of copies or trials, etc.) [2,17,29].

However, managing and enforcing stateful licenses on open platforms is difficult. Open platforms are under the control of their owners, who can attack and circumvent even sophisticated protection mechanisms by running exploits and reconfiguring the underlying operating system. Existing enforcement mechanisms have been defeated in various ways [32,33]. An attacker can easily record the platform state (e.g., hard-disks) and revert the platform to this state at a future point in time. This way he can reset a stateful license to a prior state and consequently circumvent license conditions. This can be done for instance by ordinary backup mechanisms or by applying software tools [9] that log all storage modifications to easily revoke these modifications for reuse of a license. Consequently, content providers tend to provide inflexible static licenses, which prevent users from any kind of transfer of licenses, including moving to other devices, lending or selling to other users. This approach is not tenable in the long term because its restrictiveness prevents reasonable usage scenarios like the above from being realized. Even honest users, frustrated at not being able to do what they consider reasonable, will be tempted to circumvent the license enforcement mechanisms.

Some systems attempt to augment open platforms with tamper-resistant hardware devices such as dongles [7] or smartcards [4]. Others have used closed systems [12] that consider only the providers needs The use of additional, external devices, however, cannot guarantee the integrity of the operating system and a proper behavior of applications since manipulations of the operating system or corresponding applications frequently allow users to bypass the security mechanisms.

Main Contribution: In this paper we present a security architecture and the corresponding reference implementation that enables secure enforcement of stateful licenses on open computing platforms as well as secure license transfers among

platforms. Our proposed architecture allows for protecting the security objectives of providers (license enforcement) and users (flexibility, fairer usage, and privacy). Our main goal is to show the feasibility of the legal and fairer usage allowing for transfer of licenses. To the best of our knowledge there currently exists no solution that is capable of enforcing stateful licenses on open platforms while providing security functionalities that allow to establish multilateral security. We show how our architecture can efficiently be implemented using existing virtualization and trusted computing technology.

1.1 Related Work

Shapiro and Vingralek [27] identified the replay problem in client platforms that are completely under the control of the user. The authors proposed to manage persistent states using external *locker services* or assumed a small amount of secure memory and secure one-way counters realized by battery-backed SRAM or special on-chip EEPROM/ROM functions. Tygar and Yee [37] elaborate on enforcement of static and dynamic licenses without centralized servers. They present a secure bootstrap process and protocols for sealing of data to a local and remote platforms. The proposed architecture relies on a microkernel which is running in a *physical* security partition provided by a secure coprocessor. This is different to our approach which is based on a virtualization layer offering *logical* security partitions ("compartments").

Marchesini et al. [15] use OS hardening to create "software compartments" which are isolated from each other and cannot be accessed by a "root spy". Based thereon, their design provides "compartmentalized attestation", i.e. attestation and binding of data to single compartments. Our approach does not employ OS hardening techniques to secure a complex monolithic legacy OS. Instead we put the legacy OS in a compartment which is then run on top of a virtualization layer. The performance loss is minor and the overall security improves, since the virtualization layer is much less complex than a monolithic OS kernel. Baek and Smith [6] build on Marchesini's work and implement a prototype for enforcing QoS policies on open platforms.

Publicly available documentation for common rights management systems from Microsoft [18], Authentica [5], or, Apple [2] do not mention how they resist replay attacks for their (proprietary) dynamic license implementations. Moreover, most of these solutions are closed software and cannot be verified for inherent security flaws. Some existing solutions affect the entire host security or violate user privacy [22], while others could be broken [32,33], and provide license transfers only to some selected devices. This point clearly contradicts the *first sale* concept: the licensor should be allowed to securely transfer legally obtained digital content without permission or interaction of the licensee. Other approaches [14,20] use small-value or short-term sublicenses based on a single source license to transfer rights. Since users of these systems always have full control over the platform storage, they can easily backup their (sub-)licenses and restore them after expiration. In [26], the authors propose an operating system extension that attests an integrity measurement (a SHA-1 digest over all

executed content) based on a cryptographic coprocessor. The proposed architecture allows a content provider to remotely verify the integrity of software and data of a client platform. However, this approach, reveals the user's overall platform configuration to the content provider, conflicting with the privacy principle of *least information*. Also, the content provider will only attest the last platform configuration given and is not able to predict future configuration. And even if periodic attestation was compelled, a client could still apply replay attacks between two measurements.

The *Enforcer* project [16] considers freshness by using the (non-volatile) *data integrity register* (DIR) of the TCG (Trusted Computing Group) specification version 1.1b [36]. Writing to a DIR requires owner authorization, reading can be done by anyone. Since the platform owner can still backup and restore the DIR storage, this is not secure against replay attacks.

2 System Model

2.1 Terms and Definitions

The main parties involved are *providers* (licensors) and *users* (licensees). We consider a provider as the representative party for rights-holders whereas the user represents consumers of digital content. These parties have only limited trust in each other. As shown in Figure 1, the provider distributes digital content (e.g., software, media files) together with the corresponding *license*, which defines the *usage-rights* (e.g., copy, play, print) applicable to the content. The user consumes content according to the license where the consumption is managed by the underlying platform. We distinguish two types of licenses, immutable *static licenses* and *stateful licenses* where the internal license state may change when it is used. This allows for many use cases where content consumption is somehow limited (e.g., n days or n times), or for transfer of licenses among devices.

Furthermore, we define a *compartment* as a software component that is logically isolated from other software components. *Isolation* means that these components can communicate or access each others data only over specified interfaces. The *configuration* of a compartment unambiguously describes the compartment's I/O behavior. We call the process of deriving the configuration

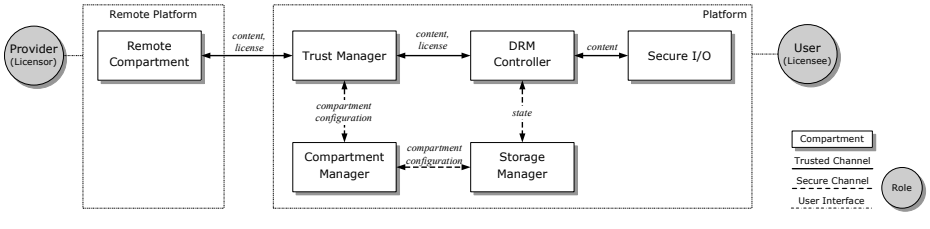


Fig. 1. System architecture

of a compartment *measurement* according to a well-defined metric. We distinguish secure and trusted communication channels between compartments. *Secure* channels ensure confidentiality and integrity of the communicated data as well as the authenticity of the endpoint compartment. A *trusted* channel is a secure channel that is bound to the configuration of the endpoint. More concretely, the channel additionally allows each endpoint compartment to (i) validate the configuration of the other endpoint compartment and (ii) to bind data to the configuration of the endpoint compartment such that solely and exclusively this compartment with this configuration can access the data. We define the *Trusted Computing Base* (TCB) as the set of all system components whose failure would allow to breach the security policy defined for the platform (e.g., as agreed by the involved parties). Note that the main design goal is to minimize the TCB.

2.2 Architecture Overview

Figure 1 gives a general overview of our architecture. The Trusted Computing Base (TCB) for our purpose (application) includes the following compartments: the Trust Manager (TM), the Storage Manager (SM), the Compartment Manager (CM), the Secure I/O (SO) compartment, and the DRM Controller (DC). Note that these components are in general distributed since all compartments communicate over trusted channels, and hence, there is no restriction on their actual physical location. In the following we briefly describe the compartments and core security properties of our architecture.

Compartment Manager (CM) initializes and closes compartments as well as *measures* compartments' configurations during initialization. Furthermore, CM enables a mapping between temporary compartment identifiers[1] and persistent compartment configurations.

Trust Manager (TM) offers basic trusted computing services and a functionality that can be used by other compartments to, e.g., establish trusted channels between compartments.

Storage Manager (SM) provides persistent storage for other compartments while preserving integrity, confidentiality, authenticity (by binding data to the compartment configuration and/or user secrets), and freshness of the stored data. Since a complete tamper-resistant storage unit would be very costly and inflexible, we used untrusted storage, i.e., a regular harddisk, with the help of TM.

Secure I/O (SO) renders (e.g., displays, plays, prints) content while preventing content leakage into untrusted compartments. Thus SO incorporates all functionality required for rendering a certain content, e.g., all corresponding drivers, rendering engines, and decoders. Moreover, access control in our architecture allows SO to communicate only with devices essential for the rendering process.

DRM Controller (DC) is a compartment that enforces the policy according to a given license attached to digital content. DC enforces security policies locally,

[1] A compartment identifier unambiguously identifies a compartment during runtime.

e.g., it uses trusted channels to decide whether a certain SO is trusted for rendering the content.[2] DC interprets the license and initiates content rendering. Moreover, DC is the core component for license transfers (cf. Section 2.3). Available content and licenses are internally indexed by DC while the index, content and licenses are persistently stored using the Storage Manager SM.

Trusted Channels as mentioned in Section 2.1, allow the involved communication end-points (compartments) to validate the configuration of the other endpoint for integrity and consequently allow determining the trustworthiness (as specified by the underlying security policy). The data sent over a trusted channel is exclusively bound to the configuration of the endpoint compartment as measured by the CM. In contrast to other approaches such as [26], which report the whole platform configuration, our architecture provides trusted channels between single compartments reducing the amount of information disclosed about the platform (privacy aspects).[3] Trusted channels can be established using the functionality offered by the Trust Manager TM. Note, we call trusted channels between compartments running on the same platform as *local* trusted channels.

Strong Isolation means runtime isolation of compartments as well as data isolation in persistent storage. Runtime isolation is provided by the underlying virtualization layer (cf. Section 3.1), and the isolation of compartments' persistent state is provided by the Storage Manager (SM).

2.3 Usage and Transfer of Licenses

In the following, we define the basic mechanisms for secure license usage and license transfers. For this we assume the following to be given, and explain in Section 3 how they are implemented: (i) strong compartment isolation (cf. Section 3.1), (ii) the proper initialization of the TCB (cf. Section 3.2), and (iii) the availability of trusted channels with freshness detection (cf. Section 3.3).

On startup, DC loads its actual content/license index i_{DC} from the Storage Manager SM using a (local) trusted channel. To provision licenses the provider establishes a trusted channel to DC. Over this channel the content and licenses are sent to DC and locally stored by SM. For *using (stateful) licenses* the user invokes DC, which loads the corresponding license, checks if all conditions for the corresponding usage-rights are fulfilled, and opens a (local) trusted channel to the secure I/O compartment SO. On the execution of the usage-right, DC updates the state of the license, synchronizes its internal state i_{DC} with the one stored by SM, decrypts the corresponding content, and invokes SO to securely render it. For *transferring stateful licenses* from a source controller DC_s to a destination controller DC_d the following steps are be taken:

[2] DC's decision is based on either a approved configuration described in the license or on the platform security policy associated with the actual TCB configuration.

[3] Further advantages of our approach are scalability and flexibility: it need not to verify the integrity of all compartments executed on the platform and the integrity verification remains valid even if the user installs or modifies other compartments since the verification is independent of other compartments running in parallel.

1. The user requests DC_s to transfer a license L to DC_d. DC_s uses TM to establish a fresh trusted channel to DC_d to send the license (and corr. content).
2. TM establishes a trusted channel with freshness detection to DC_d allowing DC_s to verify that the configuration of DC_d is conforming to the security policy of L. Note that DC_d does *not* need the equivalent verification for DC_s since the overall security architecture protects and enforces any license once accepted into any DC_s, regardless of the source of the license.[4]
3. Once the decision to transfer L to DC_d is made, DC_s invalidates L locally while synchronizing its internal state i_{DC} with SM where the identity of DC_d (e.g., a public key) is stored together with the license identity to handle possible further requests from DC_d (e.g., when the channel was disconnected for some reason). Note that freshness detection (cf. Section 3.3) will ensure that DC_d will accept L only once.
4. DC_s sends L (and corr. content) to DC_d over the fresh trusted channel. To handle transmission failures, DC_s allows retransmissions requests to DC_d.

The procedure for lending a license is similar to a license transfer: if the license allows lending DC_s generates a license for DC_d valid for the loan period, and updates the state of its own license so that it remains disabled during the loan period. This assumes the availability of secure time.

2.4 Security Objectives

We consider the following security objectives of users and providers.

(O1) *License integrity*: Unauthorized alteration of licenses must be infeasible. This is required by both provider and user.

(O2) *License enforcement*: The license can only be used according to the usage-rights prescribed by the license and to the security policy defining requirements on DC.

(O3) *Freshness*: Replay of licenses must be infeasible. Received and retrieved data is the last one sent or stored even in the case of a platform re-installation.

(O4) *Privacy:* Usage or transfer of licenses must not violate privacy policies. This concerns in particular the least information policy such that components not under full control of the user shall be able to collect, store, and reveal user's private information only to the extent required for license enforcement.

The system is not limited to a specific set of license issuers, and is capable of enforcing the terms of any license accepted by the user. Requirements like license issuance and unforgeability is considered out of the scope of this paper. Distributed authorship proofs and rights management (e.g., as in [1]) can still effectively be built based on our architecture.

[4] As fallback solution, e.g., in case of a broken DC or a dishonest provider, DC_d may also verify the validity of DC_s.

3 Reference Implementation

3.1 Overview

Our implementation primarily relies on a small security kernel, virtualization technology, and trusted computing technology. The security kernel, located as a control instance between the hardware and the application layer, implements elementary security properties like trusted channels and strong isolation between processes. Virtualization technology enables reutilization of legacy operating systems and existing applications whereas TC technology serves as root of trust. In our architecture a compartment maps to a running application or operating system, whereas a compartment configuration maps to a hash value of the software binary including all initialization information. The architecture of our implementation is shown in Figure 2 whose layers we describe below.

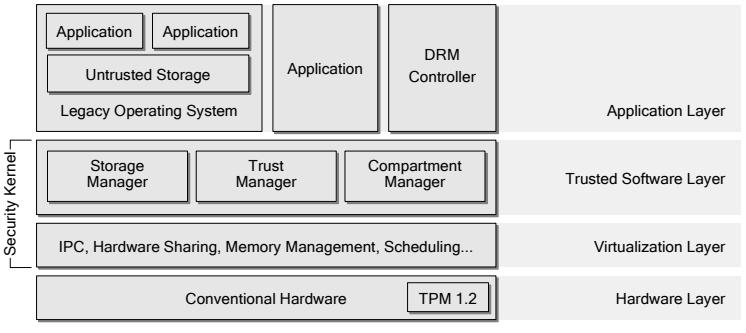


Fig. 2. Security architecture

Hardware Layer. The hardware layer consists of commercial off-the-shelf PC hardware enhanced with trusted computing technology as defined by the Trusted Computing Group (TCG) [35]. TCG has published several specification for extending the common computing platforms with cryptographic and security features in hardware and software. The main TCG specification is Trusted Platform Module (TPM) [36], which is currently implemented as dedicated cost-effective crypto chip mounted on mainboards of computing devices[5]. Many vendors already ship their platforms with TPMs (mainly laptop PCs and servers) providing the following features: A hardware-based random number generator (RNG), a cryptographic engine for encryption and signing (RSA) as well as a cryptographic hash function (SHA-1, HMAC), read-only memory (ROM) for firmware and certificates, volatile memory (RAM), non-volatile memory (EEPROM) for internal

[5] TPMs are assumed tamper-evident and will only provide a limited protection against hardware based attacks, due to the trade-off between costs and tamper protection. Nevertheless, at least rudimentary tamper precautions and tampering-detection sensors are included in the design and manufacturing process.

keys, monotonic counter values and authorization secrets, and optionally, sensors for tampering detection. Security critical operations (e.g., key generation and decryption) are performed on-chip and security critical information (e.g., secret keys) never leave the TPM unencrypted. The TPM's most important keys were the endorsement key EK, an asymmetric key that uniquely identifies each TPM; and the Storage Root Key SRK, an asymmetric key used to encrypt all other keys created by the TPM. Note that neither EK nor SRK can be read-out from the TPM. The TPM provides further a set of registers called *Platform Configuration Registers* (PCR) that can be used to store hash values.[6] During system startup, a chain of trust is established by cryptographically hashing each boot stage before execution. These hash values are also called *measurements* (in the TCG terminology) and are stored in PCRs. The set of PCR values provides is an evidence for the system's state after boot. This state is called the platform *configuration*. Based on this PCR set, among others, the two functions *sealing* resp. *binding* can be provided to relate data to a platform configuration, sealing additionally relating the data to the specific TPM instance using the TPM's endorsement key EK.

Virtualization Layer. The main task of the virtualization layer is to provide an abstraction of the underlying hardware, e.g., CPU, interrupts, devices, and to offer an appropriate management interface. Moreover, this layer enforces an access control policy based on these resources. The current implementation is based on microkernels[7] of the L4-family [13]. It implements hardware abstractions such as threads and logical address spaces as well as inter-process communication. Device drivers and other essential operating system services, such as process management and memory management, run in isolated user-mode processes. In our implementation, we kept the interfaces between layers generic to support also other virtualization technologies (e.g., Xen [25]). However, we decided to employ a L4-microkernel that allows for isolation between processes without the need to create a full OS instance in every compartment in contrast to Xen.

Trusted Software Layer. The trusted software layer, based on the PERSEUS security architecture [19], uses the functionality offered by the virtualization layer to provide security functionalities on a more abstract level. It provides elementary security properties such as trusted channels, platform policy control and compartment isolation. These realize security critical services independent of and protected from application layer compartments. The main services of the trusted software are described in Section 2.2.

Application Layer. On top of the security kernel, several instances of legacy operating systems (here Linux) as well as security-critical applications (here the DRM controller and Secure I/O) are executed in strongly isolated compartments. Unauthorized communication between compartments and unauthorized

[6] The hardware ensures that the value of a PCR can only be *extended* as follows: $PCR_{i+1} \leftarrow \mathsf{hash}[PCR_i|\mathrm{x}]$, with the previous register value PCR_i, the new register value PCR_{i+1}, and the input value x (e.g., again a hash value).

[7] A microkernel is an OS kernel that minimizes the amount of code running in privileged processor mode [21].

I/O access is prevented. The proposed architecture offers an efficient migration of existing legacy operating systems. We are currently running a para-virtualized Linux [11]. The legacy operating system provides all operating system services that are not security-critical and offers users a common environment and a large set of existing applications. If a mandatory security policy requires isolation between applications of the legacy OS, they can be executed by parallel instances of the legacy operating system.

In our reference implementation[8], DC manages, based on XrML, license interpreting and license transfers for several audio formats. Using a Linux multimedia library [10], our SO implementation provides the corresponding audio rendering and play-back.

3.2 Verifiable Initialization

For verifiable bootstrapping of the Trusted Computing Base (TCB), a TCG-enabled BIOS, called the *Core Root of Trust for Measurement* (CRTM), measures the the Master Boot Record (MBR), before passing control to it. A secure chain of measurements is then established: Before a program code is executed it is measured by a previously (measured and executed) component. For this purpose, we have modified the GRUB bootloader (cf. `www.prosec.rub.de/tgrub.html`) to measure the integrity of the TCB. The measurement results are securely stored in the PCRs of the TPM. All further compartments, applications and legacy OS instances are then subsequently loaded, measured, and executed by the Compartment Manager CM.

3.3 Trust Manager and Trusted Channels

Our Trust Manager (TM) implementation is based on the open-source TCG software stack [34]. Trusted channels can be established online or offline. The former requires a direct connection between user and provider whereas the latter does not. Examples are the online purchase of content and licenses at a provider website or obtaining content offline via DVD or as indirect copy by a third party.

Figure 3 gives a description of the protocol for establishing a trusted channel. The protocol can be decomposed into two major phases, namely issuing and verifying a target certificate, and establishing a secret key whose usage is bound to the configuration of the endpoint compartment and the underlying TCB.

If a remote compartment RC requests a trusted channel to a local compartment LC, LC passes this request to TM. TM maps LC's compartment identifier to its compartment configuration $comp\_conf_{\mathsf{LC}}$ using CM. TM then uses, by the means of TPM_CreateWrapKey[], the TPM to create a binding key pair (PK_{BIND}, SK_{BIND}) where usage of SK_{BIND} is restricted to the current TCB configuration TCB_conf measured during initialization (cf. Section 3.2). The TPM then returns PK_{BIND} and the SRK-encrypted[9] secret part ESK_{BIND}.

[8] Most of the corresponding source code is available at `www.emscb.org`

[9] The Storage Root Key (SRK) is a non-migratable key contained in the TPM as the root key for protected storage.

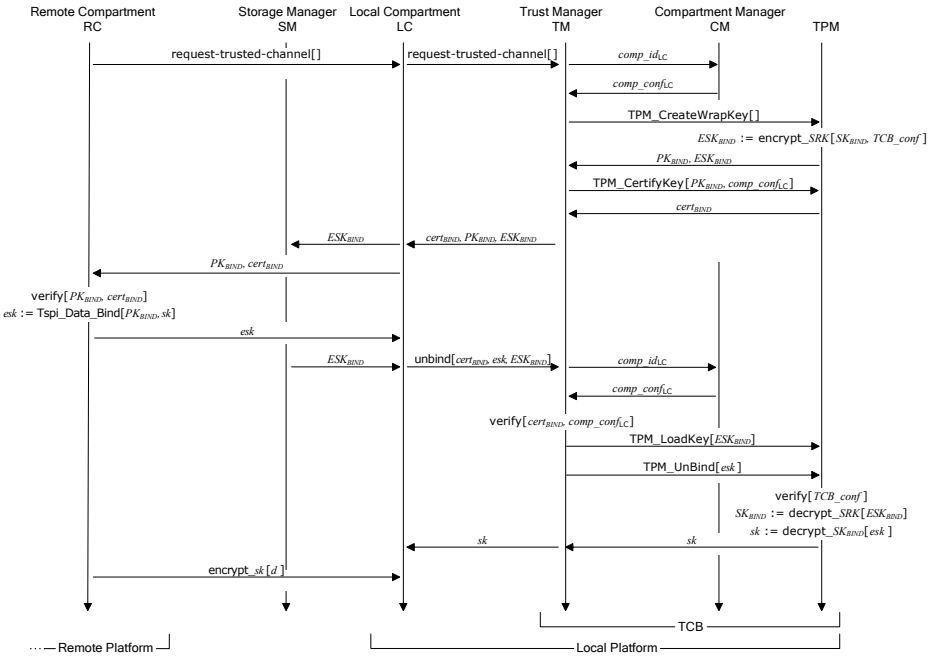


Fig. 3. Protocol for establishing a trusted channel

Then TM invokes the TPM to certify PK_{BIND} and hence to certify $comp\_conf_{LC}$ and TCB_conf using an Attestation Identity Key (AIK)[10]. We denote the result by $cert_{BIND} := \mathsf{TPM\_CertifyKey}[PK_{BIND}, comp\_conf_{LC}]$. Finally, TM returns $cert_{BIND}$ together with PK_{BIND} and ESK_{BIND} to LC. LC then stores ESK_{BIND} using SM and sends $(cert_{BIND}, PK_{BIND})$ to RC. RC verifies $cert_{BIND}$ and then the configurations $(TCB\_conf, comp\_conf_{LC})$ by comparing them with reference values (conforming to its security policy). If positive, RC generates a secret key sk and encrypts it using PK_{BIND}. The result is denoted by $esk := \mathsf{Tspi\_Data\_Bind}[PK_{BIND}, sk]$[11] and sent back to LC. Upon receipt of esk, LC loads ESK_{BIND} from SM and requests TM to unbind sk. For this, TM again requests CM for mapping LC's compartment identifier to $comp\_conf_{LC}$. Having successfully verified that $comp\_conf_{LC}$ matches the configuration denoted in $cert_{BIND}$, TM requests the TPM to unbind sk. The TPM first compares the actual PCR values to those SK_{BIND} was restricted to, before returning sk to TM. Finally, TM passes sk to LC that can now decrypt the data d (license and content) received from RC. For online trusted channels, sk is used as session key to establish a secure channel inside a subsequent server-authenticated TLS connection

[10] An AIK is a special, non-migratable, anonymized key that has been attested to come from a TCG conform platform.

[11] Tspi_Data_Bind[] is a TCG software stack function that does not require any TPM hardware (functionality).

between RC and LC[12] whereas for offline trusted channels sk is used for encryption of data before being transferred using a indirect connection between RC and LC.

Freshness extension. To tackle replay attacks we extend our trusted channels with freshness. In case of an online trusted channel, freshness can be mutually provided by the underlying TLS handshake protocol by binding the TLS channel to LC (channel binding). This can be done in various ways, e.g., by including $cert_{BIND}$ in regular TLS certificates [31]. In case of offline trusted channels (or without TLS) this can be provided by a slight protocol extension and/or measures at LC. Here different approaches are possible. A simple approach is to require LC to memorize all licenses it has received (i.e., even expired ones) to easily detect license replays. Eventually, this may amount to a huge license list, and one solution is to update (PK_{BIND}, SK_{BIND}) from time to time. Another solution is to let LC also send a nonce N together with $(cert_{BIND}, PK_{BIND})$. In the last protocol step, RC encrypts N together with corresponding data d, so that LC can verify N (and thus freshness) of d and delete N after decryption. An alternative solution to nonces is to let LC create a different public key pair (PK_L, SK_L) for each license, store SK_L in SM, and send $(PK_L, cert_{BIND}, PK_{BIND})$ to RC. Then RC encrypts data d as before using sk, but encrypts sk with PK_L *and* PK_{BIND}, i.e., $esk :=$ Tspi_Data_Bind$[PK_{BIND},$ encrypt_$PK_L[sk]]$, and sends both quantities to LC. LC now can detect replays of already known licenses by identifying PK_L. Recall that there is a unique relation between PK_L and a license. Once a license has been expired or transferred, SK_L can be deleted. In all scenarios, all secret keys and freshness verification information is persistently stored in trusted storage managed by SM (cf. Section 3.4). All solutions can defeat replays even if the platform is completely re-installed since in this case also all keys and freshness information (contained in SM) are deleted making the the corresponding licenses and content inaccessible.

We have implemented this protocol on TPMs of some major vendors (cf. [24] for more details). The TPM computation dominates the overall computation time. Hence, depending on the efficiency requirements of the underlying application, we have forseen a service (e.g., as part of the TM) that performs the related TPM tasks in software (e.g., generating binding keys). This service is clearly a part of the TCB and is included in the measurements during the verifiable initialization. In this case the trust assumptions of the TCB become stronger since the secret binding key is now in software and not in the TPM security module.

3.4 Storage Manager

The main interfaces of Storage Manager SM (cf. Figure 4) are the trusted channels load[] and store[] for loading/storing data for requesting compartments, and plain channels read[] and write[] for reading/writing data from/to an untrusted storage compartment (e.g., a hard disk drive) to persistently write respectively

[12] Alternatively, PK_{BIND} directly can be integrated into the TLS handshake, e.g., to encrypt RC's pre-master-secret.

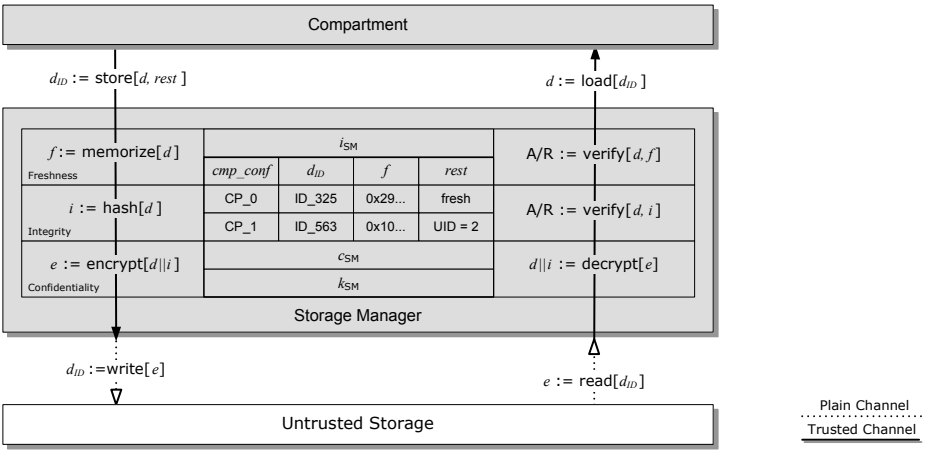


Fig. 4. Implementation of SM

read data. Internally, SM maintains an index i_{SM} for metadata of all managed data objects. The main entries in this index are: the configuration *comp_conf* of the requesting compartment, the data object identifier d_{ID}, its freshness detection information f, possible further access restrictions *rest* (e.g., user id, group id or date of expiry), a monotonic counter c_{SM} verifying the freshness of i_{SM}, and a sealed k_{SM} used to seal i_{SM} to SM's configuration.

To ensure freshness of the metadata the index i_{SM} itself, SM manages an internal software counter c_{SM} that is incremented synchronously with a TPM 1.2[13] monotonic hardware counter c_{TPM} each time SM updates its index[14]. A mismatch means outdated data which will be handled according to the underlying security policy. In order to employ TPM's monotonic counters, SM has to be initialized correctly. Figure 5 depicts the steps needed for the first initialization of SM on a new platform together with the initialization necessary for instance after rebooting the platform. At initial setup SM uses the TPM to create its internal cryptographic key k_{SM}, which is sealed to the current TCB configuration. To enable freshness detection and thus trusted storage, SM creates a monotonic counter c_{TPM} with a label *c_label* for identification and an authentication *c_auth* (e.g., a randomly chosen secret password). The initial setup finishes with the generation of i_{SM} and the sealed key ek_{SM} and writing i_{SM} (that includes c_{SM}, *c_label* and *c_auth*) encrypted on untrusted storage using k_{SM}.

After a platform reboot, SM reads the ek_{SM} from the untrusted storage and asks the TPM to unseal ek_{SM} to its internal key k_{SM}. The TPM is able to unseal k_{SM} if the platform has the same configuration as it had at the sealing process, thus preventing a modified SM to access k_{SM}. Then SM uses k_{SM} to decrypt i_{SM}

[13] TPM version 1.1b cannot be used for fresh storage [24].

[14] As specified in [36] version 1.2, a TPM supports monotonic counters with an increment rate of at least once every 5 seconds other at least 7 years.

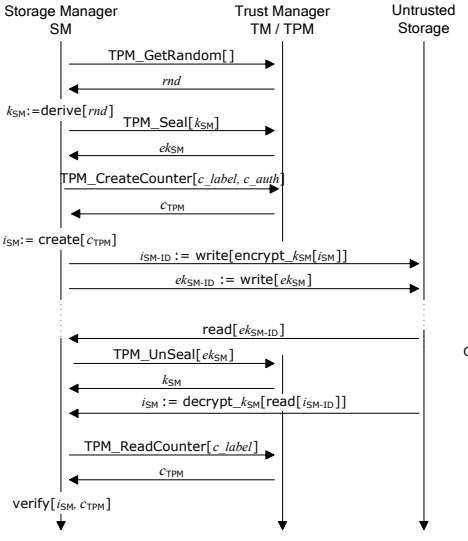


Fig. 5. SM initialization

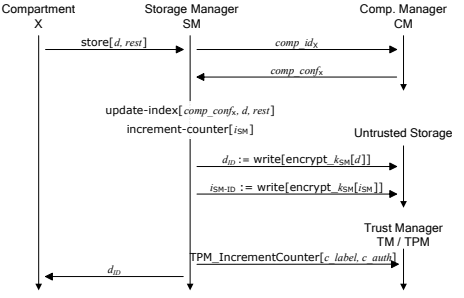


Fig. 6. SM's store protocol

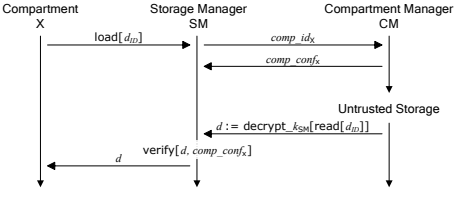


Fig. 7. SM's load protocol

and verifies freshness of i_{SM} by comparing the decrypted counter value c_{SM} of i_{SM} with the actual counter value of the corresponding hardware counter c_{TPM}.

Figure 6 depicts the protocol steps required to bind a compartment's data object (e.g., i_{DC}) persistently to its actual configuration. After the mapping of compartment identifier to the actual compartment configuration (e.g., $comp\_conf_{\mathsf{DC}}$) using CM, SM updates i_{SM} with the corresponding metadata as well as the incremented software counter c_{SM} to enable freshness detection for i_{SM}. SM encrypts both the data object and the updated index i_{SM} using k_{SM} and writes them to untrusted storage. Finally, SM synchronizes its software counter c_{SM} with the TPM's monotonic hardware counter c_{TPM} (using c_label and c_auth) and returns the data object identifier.

Figure 7 depicts the protocol steps required to load a compartment's data object. Again after a mapping of compartment identifier to the actual compartment configuration using CM, SM reads the requested data object from untrusted storage and decrypts it using k_{SM}. Before returning d to the corresponding compartment, SM verifies all existing access restrictions (e.g., freshness, or a certain user id) given on store via *rest* based on the corresponding metadata in i_{SM} and verifies that the requesting compartment has the same configuration as used on store.

4 Security Considerations

In this section we sketch the security aspects of our implementation. First we consider the core security properties (verifiable initialization, strong isolation,

trusted channels, trusted storage) provided by our implementation. Based on these properties we consider the individual security objectives (cf. Section 2.4).

Verifiable Initialization. It ensures that the TCB bootstrap is measured and securely stored in the TPM (cf. Section 3.2). Other compartments can then use TPM functionality to securely query the actual TCB configuration. Note that subsequent modifications at runtime are not reflected by the initialization measurements. However, a TCB configuration that would allow arbitrary alternation/patches of core security components cannot be considered as trustworthy.

Strong Isolation. Runtime isolation is provided by the small virtualization layer that implements only logical address spaces, inter-process communication and an appropriate interface to enforce an access control management for the underlying hardware. Device drivers and other essential operating system services, such as process management and memory management, run in isolated user-mode processes. Thus, the amount of code running in privileged ("ring 0") processor mode, is small[15] and can, in contrast to monolithic operating system kernels[16] such as Linux or MS Windows, be easier validated for correctness.

Moreover, a failure in one of these services cannot directly affect the other services, especially the code running in privileged mode. Thus, malicious device drivers cannot compromise core operating system services as they are all executed in user-mode. Isolation in persistent storage is provided by our Storage Manager (SM) implementation and the usage of trusted channels.

Trusted Channels. The establishment of a trusted channel is described detailed in Section 3.3. The inter-process communication provided by the virtualization layer enables secure channels between local compartments. Secure channels between local and remote compartments can be provided either by using the secret key *sk* to establish a secret channel inside a tunnel created by standard security protocols such as TLS [8] (online trusted channel) or by using *sk* to encrypt content at RC before sending it indirectly (e.g., via DVD or CD-ROM) to LC (offline trusted channel). As mentioned in Section 3.3 trusted channel enables access to data only by an authorized compartment (trustworthy configuration). The configuration of a compartment and the underlying TCB are securely measured during the initialization (cf. Section 3.2). Replay attacks on trusted channels can be defeated using one of the freshness solutions described in Section 3.3.

Trusted Storage. SM provides integrity, authenticity, confidentiality and freshness of data as described in Section 3.4. The integrity and confidentiality are achieved by using standard cryptographic mechanisms whereas monotonic hardware counters are used for freshness detection. We have improved common approaches while taking advantage of the strong isolation capability of our architecture that prevents the exposure of cryptographic secrets to unauthorized or malicious processes. Our SM enables compartments to persistently bind their local state to their actual configuration. The verifiable initialization (cf. Section 3.2) verifies

[15] A microkernel-based approach can be realized with around 50.000 SLOC [28].

[16] The sources lines of code (SLOC), e.g., for Windows XP are around 40 million and around 6 million for a regular Linux 2.6 kernel [30].

whether the TCB components booted are trustworthy, i.e., conform to the underlying security policy.

Given these properties we sketch the analysis of the security objectives. License integrity (O1): Trusted channels ensure that only mutually trusted compartments can modify a license, whereas strong isolation and trusted storage prevent unauthorized alteration of licenses at runtime and while persistently stored. License enforcement (O2): License and content are sent only to a local compartment whose configuration matches that of DC. Further, the isolation property prevents any other malicious code from accessing the content or modifying the license. Freshness (O3): The freshness extension (cf. Section 3.3) and SM ensure that any data loaded is the last one stored. Privacy (O4): The properties of our architecture such as isolation and binding and the fact that security policy defined by the platform owner restricts the I/O behavior of every application imply that even if third party applications, like DC, can locally enforce their own security policy, they cannot bypass the defined overall security policy. In particular, the information revealed to third parties (content providers) is restricted following the least privilege policy, e.g., only the configuration of the TCB and DC essential for transferring licenses are revealed. However, if it is required not to reveal the TCB configuration a possible extension to our architecture would be to add property-based attestation service [23] to TM and CM to hide both the (binary) configuration of the TCB and DC.

5 Summary

In this paper, we introduced the design, the realization and implementation of an open security architecture that is capable to enforce stateful licences on open platforms. Particularly, it allows the transfer of stateful licences, while preventing replay attacks. We have shown how to implement this security architecture by means of virtualization technology, an (open source) security kernel, trusted computing functionality, and a legacy operating system (currently Linux).

The building blocks needed for stateful licenses can also enable offline superdistribution [3]. For example, in our motivating scenario, Timo could generate a *new* license for Anna's device. The DRM controller will record this fact in its stateful license until Timo pays for the new copy. Allowing copies to be made while still retaining the ability for proper metering and reporting of new copies will enable rapid *legal* spread of popular content. We plan to describe this extension more fully in a forthcoming paper.

Finally, copyright itself is a strongly debated topic. In course of time, the world may develop alternative business models that do not require protection of copyright in its current form. However, the type of platform security described is also useful in many other applications like copy-protected ticketing, and electronic money. In fact, the same techniques that are used to protect the interests of a third party from a malicious device owner can also help protect the device owner from a thief who stole the device.

Acknowledgments

We thank Michael Scheibel for his contributions in jointly developing and implementing the concept of trusted channels.

References

1. Adelsbach, A., Sadeghi, A.-R., Rohe, M.: Towards multilateral secure digital rights distribution infrastructures. In: Proceedings of the ACM Workshop on Digital Rights Management (2005)
2. Apple Computer, Inc. FairPlay DRM., www.apple.com/itunes/
3. Asokan, N., Ekberg, J.-E.: Mobile digital rights management. In: Professional Mobile Internet Technical Architecture – Visions & Implementations (2002)
4. Aura, T., Gollmann, D.: Software license management with smart cards. In: Proceedings of the First USENIX Workshop on Smartcard Technology (1999)
5. Authentica, Inc. Authentica active rights management, www.authentica.com
6. Baek, K.-H., Smith, S.W.: Preventing theft of quality of service on open platforms. IEEE/CREATE-NET Workshop on Security and QoS in Communications Networks (September 2005)
7. Council, N.R.: The Digital Dilemma, Intellectual Property in the Information Age. National Academy Press, Washington, DC (2000)
8. Dierks, T., Allen, C.: RFC2246 - the TLS protocol version 1.0 (January 1999), www.ietf.org/rfc/rfc2246.txt
9. Epsilon Squared, Inc. InstallRite Version 2.5., www.epsilonsquared.com
10. Freitas, M., Roitzsch, M., Melanson, M., and Mattern, T.: The xine free multimedia player, www.xinehq.de
11. Hohmuth, M.: Linux-Emulation auf einem Mikrokern. Master's thesis, Dresden University of Technology, Dept. of Computer Science (1996)
12. Koenen, R., Lacy, J., MacKay, M., Mitchell, S.: The long march to interoperable digital rights management. Proceedings of the IEEE 92(V) (2004)
13. Liedtke, J.: Towards real microkernels. Communications of the ACM 39 (September 1996)
14. Liu, Q., Safavi-Naini, R., Sheppard, N.P.: A license-sharing scheme in digital rights management. Tech. rep., Cooperative Research Centres - Smart Internet Technology, Australia (2004)
15. Marchesini, J., Smith, S., Wild, O., Barsamian, A., Stabiner, J.: Open-source applications of TCPA hardware. In: 20th Annual Computer Security Applications Conference (2004)
16. Marchesini, J., Smith, S.W., Wild, O., MacDonald, R.: Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. Tech. Rep. TR2003-476, Dartmouth College (2003)
17. Microsoft Corporation. 60 days trial program, us1.trymicrosoftoffice.com
18. Microsoft Corporation. Windows media rights manager 10, www.microsoft.com/windows/windowsmedia/drm/default.aspx
19. Pfitzmann, B., Riordan, J., Stüble, C., Waidner, M., Weber, A.: The PERSEUS system architecture. Tech. Rep. RZ 3335 (#93381), IBM Research Division, Zurich Laboratory (April 2001)
20. Pruneda, A., Travis, J.: Metering the use of digital media content with Windows Media DRM 10, http://msdn.microsoft.com/library/en-us/dnwmt/html/meteringcontentusage10.asp

21. Robin, J.S., Irvine, C.E.: Analysis of the intel pentium's ability to support a secure virtual machine monitor. In: Proceedings of the 9th USENIX Security Symposium (2000)
22. Russinovich, M.: Sony, rootkits and digital rights management gone too far (October 2005), http://blogs.technet.com/markrussinovich/
23. Sadeghi, A.-R., Stüble, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. In: The New Security Paradigms Workshop (2004)
24. Sadeghi, A.-R., Wolf, M., Stüble, C., Asokan, N., Ekberg, J.-E.: Enabling Fairer Digital Rights Management with Trusted Computing. Tech. Rep. HGI-TR-2007-002, Horst-Görtz-Institute for IT-Security, Ruhr-University Bochum (June 2007)
25. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure hypervisor approach to trusted virtualized systems. Tech. Rep. RC23511, IBM Research Division (2005)
26. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. In: Proceedings of the 13th USENIX Security Symposium (2004)
27. Shapiro, W., Vingralek, R.: How to manage persistent state in DRM systems. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, Springer, Heidelberg (2002)
28. Singaravelu, L., Pu, C., Helmuth, C., Härtig, H.: Reducing TCB complexity for security-sensitive applications: Three case studies. In: Eurosys Conference Proceedings, Leuven, Belgium (2006)
29. Starz Entertainment Group. Video on demand service, www.vongo.com
30. Tanenbaum, A.: Keynote at linux.conf.au (January 2007)
31. TCG Infrastructure Workgroup. Tcg infrastructure workgroup subject key attestation evidence extension specification version 1.0 revision 7
32. The Hymn Project. Free your iTunes Music Store purchases from their DRM restrictions (March 2007), www.hymn-project.org
33. The Register. DVD Jon hacks Media Player file encryption (October 2005), www.theregister.co.uk/2005/09/02/dvd_jon_mediaplayer/
34. TrouSerS. The open-source TCG software stack, trousers.sourceforge.net
35. Trusted Computing Group, www.trustedcomputinggroup.org
36. Trusted Computing Group. TPM main specification. Tech. rep., www.trustedcomputinggroup.org/specs/TPM/
37. Tygar, J., Yee, B.: Dyad: a system using physically secure coprocessors. In: Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment (1994)

Traitor Tracing with Optimal Transmission Rate

Nelly Fazio[1,⋆], Antonio Nicolosi[2,⋆], and Duong Hieu Phan[3,⋆⋆]

[1] IBM Almaden Research Center
nfazio@us.ibm.com
[2] New York University and Stanford University
nicolosi@scs.stanford.edu
[3] France Telecom R&D and University of Paris 8
duonghieu.phan@orange-ftgroup.com

Abstract. We present the first traitor tracing scheme with efficient black-box traitor tracing in which the ratio of the ciphertext and plaintext lengths (the *transmission rate*) is asymptotically 1, which is optimal. Previous constructions in this setting either obtained constant (but not optimal) transmission rate [16], or did not support black-box tracing [10]. Our treatment improves the standard modeling of black-box tracing by additionally accounting for pirate strategies that attempt to escape tracing by purposedly rendering the transmitted content at lower quality.

Our construction relies on the *decisional bilinear Diffie-Hellman* assumption, and attains the same features of public traceability as (a repaired variant of) [10], which is less efficient and requires non-standard assumptions for bilinear groups.

Keywords: Traitor Tracing, Constant Transmission Rate, Fingerprint Codes, Bilinear Maps.

1 Introduction

Traitor tracing schemes constitute a very useful tool against piracy in the context of digital content distribution. They are multi-recipient encryption schemes that can be employed by *content providers* that wish to deliver copyrighted material to an exclusive set of users. Each user holds a decryption key that is fingerprinted and bound to his identity. If a group of subscribers (the *traitors*) collude to construct an illegal device (the *pirate decoder*), the *security manager* can run a specialized traitor tracing algorithm to uncover the source of the leakage. Therefore, a traitor tracing scheme deters subscribers of a distribution system from leaking information by the mere fact that the identities of the leaking entities can then be revealed.

The first formal definition of traitor tracing scheme appears in Chor *et al.* [11,12], whose construction requires storage and decryption complexity $O(t^2 \log^2 t \log(n/t))$ and communication complexity $O(t^3 \log^4 t \log(n/t))$, where n is the size of the universe of users and t is an upper bound on the number of traitors. Stinson and Wei later suggested in [22] explicit combinatorial construction that achieve better efficiency for small values of t and n.

⋆ Research conducted in part while visiting ENS, Paris and Stanford University, CA.
⋆⋆ Research conducted in part while at ENS, Paris and University College London.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 71–88, 2007.

The work of [19,12] introduced the notion of *threshold* traitor tracing scheme, where the tracing algorithm is only required to guarantee exposure of the traitors' identities for pirate decoders whose decryption probability is better than a given threshold β. The scheme of [19] achieves storage complexity $O(t/\beta \log(t/\varepsilon))$, where ε is the probability of successfully tracing one of the traitors. Moreover, the scheme has communication complexity linear in t and constant decryption complexity.

In [4], Boneh and Franklin present an efficient public-key traitor tracing scheme with deterministic t-tracing based on an algebraic approach. Its communication, storage and decryption complexities are all $O(t)$. The authors also introduce the notion of *non-black-box traceability*: given a "valid" key extracted from a pirate device (constructed using the keys of at most t users), recover the identity of at least one traitor. This is in contrast with the notion of *black-box tracing* (on which we focus in this paper), where the traitor's identity can be uncovered by just observing the pirate decoder's replies on "well crafted" ciphertexts. More recently, Boneh *et al.* [5,7] proposed traitor tracing schemes that withstand any number of traitors (*full traceability*), while requiring a sub-linear ciphertext length ($O(\sqrt{n})$).

Constant Transmission Rate. As pointed out by Kiayias and Yung in [16], an important problem in designing practical traitor tracing schemes is to ensure a low *transmission rate*, defined as the asymptotic ratio of the size of ciphertexts over the size of plaintexts, while at the same time minimize the *secret-* and the *public-storage* rates, similarly defined as the asymptotic ratio of the size of user-keys and of public-keys over the size of plaintexts.[1] Under this terminology, the transmission rate of all the above mentioned solutions is linear w.r.t. the maximal number t of traitors, whereas in [16], Kiayias and Yung show that if the plaintexts to be distributed are large (which is the case for most applications of traitor tracing, such as distribution of multimedia content), then it is possible to obtain ciphertexts with constant expansion rate. Their solution is based on collusion-secure fingerprint codes [6,23] and its parameters are summarized in Figure 1.

Besides the clear benefit in terms of communication efficiency, schemes with constant transmission rate also enjoy efficient black-box traceability, while schemes with linear transmission rate are inherently more limited in this regard [15] (*e.g.,* the black-box traitor tracing of [4] takes time proportional to $\binom{n}{t}$).

In [10], Chabanne *et al.* extend the setting of [16] with the notion of *public traceability*: Whereas traditional tracing algorithms require knowledge of the system's secret information, in a scheme with public traceability everyone can run the tracing algorithm. In this paper, we also consider *local public traceability*, whereby public information suffices to carry out the preliminary phase of tracing, which requires interaction with the pirate decoder, and results in an encoding of the traitor's identity that can be decoded with a master key. This separation of tasks ensures that the system's secret information

[1] We adopt a terminology slightly different from the one of [16], which uses the term *ciphertext/user-key/public-key* rates, for what we called *transmission/secret-storage/public-storage* rates. Moreover, in [16] *transmission rate* refers to the sum of the all the three rates. Our choice is of course mostly a matter of taste: we prefer the terminology of this paper as it makes more evident the role played by each quantity in a concrete implementation of the system.

	Trans. Rate	S-Storage Rate	P-Storage Rate	BB Tracing	Public Traceability	Hardness Assumption
BF[4]	$2t+1$	$2t$	$2t+1$	$\times$	$\times$	DDH
KY[16]	3	2	4	$\surd*$	$\times$	DDH
CPP[10]	1	2	1	$\times$	$\times$	DBDH^2-E $\wedge$ DBDH^1-M
PST[20]	7	1	1	$\surd$	full	DDH
Repaired CPP	3	2	6	$\surd$	local	DBDH^2-E $\wedge$ DBDH^1-M
Our Scheme	1	2	10	$\surd$	local	DBDH

Fig. 1. Comparison of rates (*transmission*, *secret-* and *public-storage* rates) and tracing features (*black-box tracing* and *public traceability*) between existing schemes and our construction. The "*" in the row labeled "KY" refers to the fact that the scheme of [16] can support black-box tracing using the tracing algorithm that we describe in the full version [13] The row labeled "PST" refers to instantiating their generic construction with ternary IPP codes and ElGamal-style encryption. The row labeled "Repaired CPP" refers to the variant of the scheme of [10] that we suggest in the full version [13] to support black-box tracing.

is only needed for off-line operations (*i.e.*, user registration and possibly the final phase of tracing), thus improving the overall security of the system by allowing for safer storage solutions.

The work of [20] describes a traitor tracing scheme with constant (but not optimal) transmission rate and (full) public traceability based on Identifiable Parent Property (IPP) codes. Figure 1 also reports on these two schemes. One could think that traitor tracing schemes with linear transmission rate (*e.g.* [4]) could easily be turned into schemes with constant transmission rate by means of hybrid encryption: To send a large message, pick a random session key, encrypt it with the given traitor tracing scheme, and append a symmetric encryption of the message under the chosen anonymous session key. This approach, however, suffers from a simple yet severe untraceable pirate strategy: Just decrypt the session key and make it available to the "customers" on the black market, *e.g.,* via anonymous e-mail, or via text-messaging from a pre-paid cellphone. Clearly, when a traitor tracing scheme is used to encrypt the content directly, this "re-broadcasting" strategy becomes much less appealing for would-be pirates, because of the higher costs and exposure risks associated with running a high-bandwidth darknet.

Our Contributions. We present the first public-key traitor tracing scheme with efficient black-box traitor tracing and local public traceability in which the transmission rate is asymptotically 1, which is optimal. Encryption involves the same amount of computation as in [10]; decryption is twice as fast. We also considerably simplify the computational hardness requirements, relying just on the DBDH assumption—much weaker and more widely accepted than the non-standard bilinear assumptions employed in [10].

Our treatment improves the standard modeling of black-box tracing by additionally accounting for pirate strategies that attempt to escape tracing by purposedly rendering the transmitted content at lower quality (*e.g.* by dropping every other frame from the decrypted video-clip, or skipping few seconds from the original audio file).

As additional contribution, we point out and resolve an issue in the black-box tracing of [16] (which was also independently addressed in a revised version of their work [17]).

We then show that [10], which extends [16] and inherits its tracing mechanism, inherits in fact the above-mentioned problem, too. In this case, however, fixing the black-box functionality requires changes that intrinsically conflict with the optimizations put up by [10] to achieve optimal transmission rate. In other words, [10] can either provide optimal transmission rate with only non-black-box tracing, or support local public traceability with sub-optimal transmission rate, but cannot achieve both at the same time.

A Comparison with [5,7]. The schemes of [5,7] are the most efficient ones supporting full collusion, but they are not well suited for the more practical case of small number of traitors (say, logarithmic in the size of the entire user population). Indeed, in this case, the ciphertext in the schemes of [5,7] still contains $O(\sqrt{n})$ elements. In our scheme, assuming the number of traitors t is logarithmic in the number of users n, the ciphertext has poly-logarithmic length $v = O(t^2(\log n + \log \frac{1}{\varepsilon})) = O(\log^3 n)$, which is asymptotically superior to the $O(\sqrt{n})$-ciphertexts of [5,7]. More importantly, the tracing algorithms of [5,7] require $O(n^2)$ decryption queries to the pirate decoder, whereas our scheme employs $O(v) = O(\log^3 n)$ decryption queries, and is moreover completely parallelizable.

2 Preliminaries

The security properties of our construction hinge upon the decisional bilinear Diffie-Hellman assumption (DBDH) for $(\mathcal{G}_1, \mathcal{G}_2)$. We refer the reader to the full version of this paper [13] for the relevant definitions.

Collusion-Secure Codes. Collusion-secure codes [6] provide a powerful tool against illegal redistribution of fingerprinted material in settings satisfying the following *Marking Assumption*: 1) it is possible to introduce small changes to the content at some discrete set of locations (the *marks*), while preserving the "quality" of the content being distributed; but 2) it is infeasible to apport changes to a mark without rendering the entire content "useless" *unless* one possesses two copies of the content that differ at that mark. Below, we include a formalization of the notion of collusion-secure codes, adapted from [6].

Definition 1. *Let Σ be a finite alphabet, and $n, v \in \mathbb{Z}_{\geq 0}$. An (n, v)-code over Σ is a set of n v-tuples of symbols of Σ: $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\} \subseteq \Sigma^v$.*

Definition 2. *Let T be a subset of indices in $[1, n]$. The set of undetectable positions for T is: $R_T = \{\ell \in [1, v] \mid (\forall i, j \in T).[\omega_\ell^{(i)} = \omega_\ell^{(j)}]\}$.*

Notice that for each $i \in T$, the projection of each codeword $\omega^{(i)}$ over the undetectable positions for T is the same; we denote this common projected sub-word as $\omega_{|R_T}$. By the Marking Assumption, any "useful" copy of the content created by the collusion of the users in T must result in a tuple $\bar{\omega}$ whose projection over R_T is also $\omega_{|R_T}$. This is captured by the following:

Definition 3. *The set of feasible codewords for T is: $F_T = \{\bar{\omega} \in (\Sigma \cup \{?\})^v \mid \bar{\omega}_{|R_T} = \omega_{|R_T}\}$.*

Definition 4. *Let $\varepsilon > 0$ and $t \in \mathbb{Z}_{\geq 0}$. $\mathcal{C}$ is an (ε, t, n, v)-collusion-secure code over Σ if there exists a probabilistic polynomial-time algorithm $\mathcal{T}$ such that for all $T \subseteq [1, n]$ of size $\mid T \mid \leq t$, and for all $\bar{\omega} \in F_T$, it holds that: $\Pr[\mathcal{T}(r_{\mathcal{C}}, \bar{\omega}) \in T] \geq (1 - \varepsilon)$, where the probability is over the random coins $r_{\mathcal{C}}$ used in the construction of the (n, v)-code $\mathcal{C}$, and over the random coins of $\mathcal{T}$.*

3 Public-Key Traitor Tracing Scheme with Public Traceability

Definition 5 (Public-Key Traitor Tracing Scheme). *A public-key traitor tracing scheme is a 5-tuple of probabilistic polynomial-time algorithms* (**Setup**, **Reg**, **Enc**, **Dec**, **Trace**)*, where:*

Setup: *On input a security parameter 1^κ, a collusion threshold 1^t, and a bound n on the maximum number of users, returns a public key* pk *along with some master secret information* msk *(*cf. **Reg** *and* **Trace***);*

Reg: *Given* msk *and a user index $i \in [1, n]$, outputs a "fingerprinted" user key* sk_i*;*[2]

Enc: *On input key* pk *and a message m (from the appropriate message space $\mathcal{M}$, implicitly described by* pk*), returns a (randomized) ciphertext ψ;*

Dec: *On input a user key* sk_i *and a ciphertext ψ, recovers the message encrypted within ψ;*

Trace: *Given the master secret key* msk*, the public key* pk*, and black-box access to a "pirate" decoder capable of inverting the* $\mathbf{Enc}(\mathsf{pk}, \cdot)$ *functionality, returns the user index of one of the traitors that contributed his/her user key for the realization of the pirate decoder, or the special user index 0 upon failure.*

For correctness, decryption with any user key output by **Reg** *should "undo" encryption,* i.e.*,* $\mathbf{Dec}(\mathsf{sk}_i, \mathbf{Enc}(\mathsf{pk}, m)) = m$.

Definition 6 (Full/Local Public Traceability). *A public-key traitor tracing scheme is said to support: 1)* public traceability *if the* **Trace** *algorithm can be implemented without the master secret key* msk*; or 2)* local public traceability *if the* **Trace** *algorithm can be split in an on-line phase, in which the pirate decoder can be queried without knowledge of the secret key, and an off-line phase, without access to the pirate decoder, that can retrieve the identity of the traitor from the master secret key and the output of the publicly executable on-line phase.*

Definition 7 (Indistinguishability under Chosen-Plaintext Attack). *A public-key traitor tracing scheme satisfies $\varepsilon_{\mathrm{ind}}$-indistinguishability if, for any pair of probabilistic polynomial-time algorithms $(\mathcal{A}_1, \mathcal{A}_2)$, it holds that:*

$$\Pr\left[\mathcal{A}_2(\mathsf{state}, \psi^*) = b^* \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \xleftarrow{R} \mathbf{Setup}(1^\kappa, 1^t, n),\\ (m_0, m_1, \mathsf{state}) \xleftarrow{R} \mathcal{A}_1(\mathsf{pk}),\\ b^* \xleftarrow{R} \{0,1\}, \psi^* \xleftarrow{R} \mathbf{Enc}(\mathsf{pk}, m_{b^*})\end{array}\right] \leq \frac{1}{2} + \varepsilon_{\mathrm{ind}},$$

where the probability is over b^*, and the random coins of $\mathcal{A}_1$, $\mathcal{A}_2$, **Setup***, and* **Enc***.*

[2] Equivalently, we can think of **Setup** as outputting a vector of user keys, one per each user in the system; we will refer to either formalization interchangeably.

Requirements on the Tracing Functionality. Existing literature usually models black-box traceability as the ability to "extract" the identity of (at least) one traitor from pirate decoders that *correctly* invert the decryption algorithm (under appropriate efficiency and success probability constraints). This approach, however, is often criticized because it leaves the way open for pirate decoders that decrypt ciphertexts into plaintexts that closely resemble (but are not identical to) the original plaintexts. To account for pirate strategies of this sort, we allow traitors to specify a notion of "resemblance" in the form of a polynomial-time reflexive, symmetric binary relation $\mathcal{R}$ over plaintexts, with $\mathcal{R}(m, m') = 1$ if customers would accept m' as a reasonable replacement for m.[3] The only semantic constraint on $\mathcal{R}$ is that it shall not be so lax as to deem random[4] plaintexts similar to fixed ones, *i.e.*, the quantity $p_{\mathcal{R}} \doteq \max_{m \in \mathcal{M}} \Pr[\mathcal{R}(m, m') = 1 \mid m' \xleftarrow{R} \mathcal{M}]$ shall be negligible (otherwise tracing is impossible, since a keyless decoder could simply output a random plaintext as a "reasonable" decryption of any ciphertext). Similarly, tracing needs only be effective against efficient decoders $\mathcal{D}$ whose success probability $p_{\mathcal{D}} \doteq \Pr[\mathcal{R}(m, \mathcal{D}(\mathbf{Enc}(\mathsf{pk}, m))) = 1 \mid m \xleftarrow{R} \mathcal{M}]$ is non-negligible.

Definition 8. *A public-key traitor tracing scheme is $\varepsilon_{\mathrm{trac}}$-traceable if for any probabilistic polynomial-time traitor strategy $\mathcal{A}$, it holds that:*

$$\Pr\left[\mathbf{Trace}^{\mathcal{D}(\cdot)}(\mathsf{pk}, \mathsf{msk}) \notin T \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \xleftarrow{R} \mathbf{Setup}(1^\kappa, 1^t, n), \\ (\mathcal{D}, \mathcal{R}) \xleftarrow{R} \mathcal{A}(\mathsf{pk})^{\mathbf{Reg}(\mathsf{msk}, \cdot)}\end{array}\right] \leq \varepsilon_{\mathrm{trac}}$$

where $\mathcal{M}$ is the message space, $T \subseteq [1, n]$ is the set of up to t indices on which $\mathcal{A}$ queried the $\mathbf{Reg}(\mathsf{msk}, \cdot)$ oracle, $\mathcal{D}$ and $\mathcal{R}$ both run in probabilistic polynomial-time and are such that $p_{\mathcal{D}}$ is non-negligible and $p_{\mathcal{R}}$ is negligible, and the probability is over the coins of $\mathbf{Setup}$*,* $\mathbf{Reg}$*, $\mathcal{A}$, $\mathcal{D}$ and* $\mathbf{Trace}$*.*

Notice that Definition 8 subsumes the case that the traitor strategy $\mathcal{A}$ only produces a "good" pirate decoder $\mathcal{D}$ with a low (but non-negligible) probability: indeed, any such strategy can be "boosted" by simply keeping executing $\mathcal{A}$ on fresh random coins, until the pirate decoder $\mathcal{D}$ that $\mathcal{A}$ outputs is a good one (which can be efficiently tested by estimating $\mathcal{D}$'s decryption capability on the encryption of a random plaintext).

4 Public-Key Traitor Tracing with Public Traceability, Black-Box Tracing and Optimal Transmission Rate

Similarly to the schemes of [16] and [10], our construction is based on the use of an (ε, t, n, v)-collusion-secure code $\mathcal{C}$ over the alphabet $\{0, 1\}$ (*cf.* Definition 4). At a high level, the idea is to first define a two-user sub-scheme resilient against a single traitor, and then "concatenate" v instantiations of this sub-scheme according to the code $\mathcal{C}$. Although the overall architecture that we follow is well-known, achieving optimal transmission rate along these lines requires solving a number of technical problems, on which we elaborate in Section 4.4.

[3] Alternatively, the resemblance relation $\mathcal{R}$ could be specified as a parameter of the scheme in the definition of the $\mathbf{Trace}$ algorithm.

[4] For the sake of simplicity, in this paper we discuss only the case of random sampling from $\mathcal{M}$, but the treatment generalizes to the case of other plaintext distribution with high min-entropy.

4.1 Our Two-User Sub-Scheme

Setup: Given a security parameter 1^κ, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ of order q, and an admissible bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$. Choose an arbitrary generator $P \in \mathcal{G}_1$.

Step 2: Pick random elements $a, b, c \in \mathbb{Z}_q^*$, and set $Q \doteq aP, R \doteq bP, h \doteq e(P, cP)$. Compute two linearly independent vectors (α_0, β_0) and (α_1, β_1) in $\mathbb{Z}_q$ such that $b\alpha_\sigma + a\beta_\sigma = c \bmod q$, for $\sigma \in \{0, 1\}$. The private key of the security manager is set to be $\mathsf{msk} \doteq (a, b, \alpha_0, \beta_0, \alpha_1, \beta_1)$.

Step 3: For $\sigma \in \{0, 1\}$, let $A_\sigma \doteq \alpha_\sigma R$ and $B_\sigma \doteq \beta_\sigma P$. Choose a universal hash function $H : \mathcal{G}_2 \to \{0, 1\}^\kappa$, and set the public key of the scheme to be the tuple

$$\mathsf{pk} \doteq (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P, Q, R, A_0, B_0, A_1, B_1).$$[5]

The associated message space is $\mathcal{M} \doteq \{0, 1\}^\kappa$.

Reg: For $\sigma \in \{0, 1\}$, the secret key of user σ is set to be $\mathsf{sk}_\sigma \doteq \alpha_\sigma$. Notice that $cP = \alpha_\sigma R + \beta_\sigma Q$ and hence $h = e(P, cP) = e(P, \alpha_\sigma R) \cdot e(Q, \beta_\sigma P) = e(P, A_\sigma) \cdot e(Q, B_\sigma)$, for $\sigma \in \{0, 1\}$.

Enc: Given pk, anybody can encrypt a message $m \in \mathcal{M}$ by first selecting a random $k \in \mathbb{Z}_q$ and then creating the ciphertext $\psi \doteq \langle U, V, W \rangle \in \mathcal{G}_2 \times \mathcal{G}_1 \times \mathcal{M}$ where

$$U \doteq e(P, R)^k, \quad V \doteq kQ, \quad W \doteq m \oplus H(h^k)$$

Dec: Given a ciphertext $\psi = \langle U, V, W \rangle$, user σ computes $h^k = U^{\alpha_\sigma} \cdot e(V, B_\sigma)$ and recovers $m = W \oplus H(h^k)$. Correctness of the decryption algorithm is clear by inspection.

Trace: To trace a decoder $\mathcal{D}$ with resemblance relation, feed $\mathcal{D}$ with the "illegal" ciphertext $\hat{\psi} \doteq \langle e(P, R)^{k'}, kQ, \hat{m} \oplus H(e(P, A_\sigma)^{k'} e(Q, B_\sigma)^k) \rangle$, for random $\sigma \in \{0, 1\}$, $k, k' \in \mathbb{Z}_q$, $\hat{m} \in \mathcal{M}$. If the output m^* of $\mathcal{D}$ satisfies $\mathcal{R}(\hat{m}, m^*) = 1$, then return σ as the traitor's identity; otherwise, pick fresh random $\sigma \in \{0, 1\}, k, k' \in \mathbb{Z}_q, \hat{m} \in \mathcal{M}$ and repeat.

Before moving on to the security and traceability of our two-user scheme in the sense of Definitions 7 and 8 (*cf. Section* 3), we remark that **Trace** does not require knowledge of the master secret key msk, and thus it supports full public traceability (*cf.* Definition 6). Also, notice that decryption requires only one pairing computation.

4.2 Indistinguishability Under Chosen-Plaintext Attack

Theorem 9. *Under the* DBDH *assumption for* $(\mathcal{G}_1, \mathcal{G}_2)$*, the scheme in Section 4.1 is secure w.r.t. indistinguishability under chosen-plaintext attack (*cf. *Definition 7).*

[5] Note that there is no need to explicitly include h in the public key, as it can be derived as $h = e(P, A_\sigma) \cdot e(Q, B_\sigma)$. Caching the value of h, however, is recommendable when public storage is not at a premium, as that would save two pairing computations during encryption.

Proof. To a contradiction, let us assume that the scheme does not satisfy Definition 7 *i.e.*, there is an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that, given the public key $\mathsf{pk} = (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P, Q, R, A_0, B_0, A_1, B_1)$, can break the scheme with non-negligible advantage ε_{ind}. We then construct an algorithm $\mathcal{B}$ (whose running time is polynomially related to $\mathcal{A}$'s) that breaks the DBDH assumption with probability $\varepsilon_{\mathsf{DBDH}} = \varepsilon_{\text{ind}}$.

Algorithm $\mathcal{B}$ is given as input an instance (P', xP', yP', zP', h') of the DBDH problem in $(\mathcal{G}_1, \mathcal{G}_2)$; its task is to determine whether $h' = e(P', P')^{xyz}$, or h' is a random element in $\mathcal{G}_2$. $\mathcal{B}$ proceeds as follows:

Setup: $\mathcal{B}$ sets $P \doteq xP'$ and $Q \doteq P'$. Then, $\mathcal{B}$ picks $r \overset{R}{\leftarrow} \mathbb{Z}_q^*$, and sets $R \doteq rQ$. $\mathcal{B}$ now chooses $\beta_0, \beta_1 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and computes $B_0 \doteq \beta_0 P$ and $B_1 \doteq \beta_1 P$. Then, $\mathcal{B}$ sets $A_0 \doteq zP'$ and $h \doteq e(P, A_0) \cdot e(Q, B_0)$. Finally, $\mathcal{B}$ sets $A_1 \doteq A_0 + \beta_0 Q - \beta_1 Q$, so that in fact $h = e(P, A_\sigma) \cdot e(Q, B_\sigma)$, for $\sigma \in \{0, 1\}$, as required.
$\mathcal{B}$ can now set $\mathsf{pk} \doteq (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P, Q, R, A_0, B_0, A_1, B_1)$ and send it to $\mathcal{A}_1$.

Challenge: $\mathcal{A}_1$ outputs two messages m_0, m_1 on which it wishes to be challenged, along with some state state to be passed to $\mathcal{A}_2$. To prepare the ciphertext, $\mathcal{B}$ picks random $b^* \in \{0, 1\}$, and sets

$$U \doteq e(P, yP')^r (= e(P, R)^y), V \doteq yP' (= yQ), W \doteq m_{b^*} \oplus H(h' \cdot e(yP', xP')^{\beta_0}).$$

(Notice that this implicitly defines $k = y$.) Then, $\mathcal{B}$ sends $\mathcal{A}_2$ the challenge ciphertext $\psi^* \doteq (U, V, W)$, along with the state information state.

Guess: Algorithm $\mathcal{A}_2$ outputs a guess $b' \in \{0, 1\}$. $\mathcal{B}$ returns 1 if $b' = b^*$ and 0 otherwise.

If $h' = e(P', P')^{xyz}$, then $\mathcal{A}_2$ gets a valid encryption of m_{b^*}, since (as we verify below) in this case the input to the hash function in the computation of W is just h^k:

$$\begin{aligned} h' \cdot e(yP', xP')^{\beta_0} &= e(P', P')^{xyz} \cdot e(yP', \beta_0(xP')) = e(xP', zP')^y \cdot e(P', \beta_0(xP'))^y \\ &= e(P, A_0)^y \cdot e(Q, B_0)^y = [e(P, A_0) \cdot e(Q, B_0)]^y = h^y = h^k, \end{aligned}$$

as required by the encryption algorithm. Therefore, in this case $\mathcal{A}$ will successfully guess $b' = b^*$ with probability $\varepsilon_{\text{ind}} + 1/2$.

On the other hand, when h' is a random element of $\mathcal{G}_2$, the input to H is a random value, independent of any other information in the adversary's view. Since H is chosen from a universal hash function family, its output is also (almost) uniformly random in $\{0, 1\}^\kappa$, so that the value of W (and hence the whole challenge ciphertext ψ^*) is completely independent from m_{b^*}. Thus, in this case $b' = b^*$ holds with probability $1/2$.

It follows that adversary $\mathcal{B}$ breaks the DBDH assumption with non-negligible advantage $\varepsilon_{\mathsf{DBDH}} = \varepsilon_{\text{ind}}$, contradicting our hardness assumption. □

4.3 Traceability

To assess the effectiveness of the **Trace** algorithm from Section 4.1, we start with some observations about the illegal ciphertexts that **Trace** uses in querying the decoder $\mathcal{D}$:

Definition 10 (Valid and Probe Ciphertexts). *Let $\sigma \in \{0, 1\}$, $\hat{m} \in \mathcal{M}$, $\hat{U} \in \mathcal{G}_1$, $\hat{V} \in \mathcal{G}_2$, $\hat{W} = \hat{m} \oplus H(\hat{U}^{\alpha_\sigma} e(\hat{V}, B_\sigma))$, and $\hat{\psi} = \langle \hat{U}, \hat{V}, \hat{W} \rangle$. We say that the ciphertext $\hat{\psi}$ is:*

– ***valid****, if* $\hat{U} = e(P,R)^k, \hat{V} = kQ$*, for some* $k \in \mathbb{Z}_q$*;*
– σ***-probe****, if* $\hat{U} = e(P,R)^{k'}, \hat{V} = kQ$*, for distinct* $k, k' \in \mathbb{Z}_q$*.*

Lemma 11 (Indistinguishability of Valid *vs.* Probe Ciphertexts). *Under the* DBDH *assumption for* $(\mathcal{G}_1, \mathcal{G}_2)$*, given the public key* $\mathsf{pk} = (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P, Q, R, A_0, B_0, A_1, B_1)$ *and the secret key* $\mathsf{sk}_\tau = \alpha_\tau$ *of user* $\tau \in \{0,1\}$ *(where* $A_\tau = \alpha_\tau R$*), it is infeasible to distinguish a valid ciphertext from a* τ*-probe.*

Proof. For simplicity, assume $\tau = 0$. We proceed by contradiction: assume there is an adversary $\mathcal{A}$ that, given the public key $\mathsf{pk} = (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P, Q, R, A_0, B_0, A_1, B_1)$ and the secret key α_0 of user 0, can distinguish valid ciphertexts from probes with probability ε. We then construct an algorithm $\mathcal{B}$ (whose running time is polynomially related to $\mathcal{A}$'s) that breaks the DBDH assumption with probability $\varepsilon_{\mathsf{DBDH}} = \varepsilon$.

Algorithm $\mathcal{B}$ is given as input an instance (P', xP', yP', zP', h') of the DBDH problem in $(\mathcal{G}_1, \mathcal{G}_2)$; its task is to determine whether $h' = e(P',P')^{xyz}$ or h' is a random element in $\mathcal{G}_2$. $\mathcal{B}$ proceeds as follows:

Setup: $\mathcal{B}$ lets $P \doteq xP'$, $Q \doteq P'$, $R \doteq yP'$, chooses $\alpha_0, \beta_0, \beta_1 \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$ and computes $A_0 \doteq \alpha_0 R$, $B_0 \doteq \beta_0 P$ and $B_1 \doteq \beta_1 P$. $\mathcal{B}$ also sets $A_1 \doteq A_0 + \beta_0 Q - \beta_1 Q$, which implicitly defines $h = e(P, A_0) \cdot e(Q, B_0) = e(P, A_1) \cdot e(Q, B_1)$. $\mathcal{B}$ now defines $\mathsf{pk} \doteq (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P, Q, R, A_0, B_0, A_1, B_1)$. Then, $\mathcal{B}$ prepares a challenge ciphertext $\hat{\psi} \doteq \langle \hat{U}, \hat{V}, \hat{W} \rangle$ by setting $\hat{U} \doteq h'$, $\hat{V} \doteq zP' (= zQ$, thus implicitly defining $k = z)$ and $\hat{W} \doteq \hat{m} \oplus H(\hat{U}^{\alpha_0} e(\hat{V}, B_0))$, for $\hat{m} \stackrel{R}{\leftarrow} \mathcal{M}$. At this point, $\mathcal{B}$ feeds $\mathcal{A}$ with pk, $\hat{\psi}$, and α_0.

Attack: $\mathcal{A}$ returns her guess to whether $\hat{\psi}$ is a valid ciphertext or a probe (w.r.t. the public key pk).

Break: $\mathcal{B}$ outputs yes or no accordingly.

If $h' = e(P',P')^{xyz}$, then $\mathcal{A}$ gets a valid ciphertext since $h' = e(xP', yP')^z = e(P,R)^z$, consistently with the value of $\hat{V} = zQ$, as required by the encryption algorithm. Otherwise, h' is a random value in $\mathcal{G}_2$, of the form $h' = e(P,R)^{k'}$, for some k' totally independent from $k = z$, and thus $\hat{\psi}$ is a 0-probe. Therefore, $\mathcal{B}$ breaks the DBDH assumption with the same advantage as $\mathcal{A}$'s *i.e.*, $\varepsilon_{\mathsf{DBDH}} = \varepsilon$. □

An important consequence of Lemma 11 is that pirate decoders created by user τ reply to τ-probes with an m^* such that $\mathcal{R}(\hat{m}, m^*) = 1$ with non-negligible probability $\hat{p}_\mathcal{D}$:

Corollary 12. *Let* $\mathcal{D}$*,* $\mathcal{R}$ *be the pirate decoder and resemblance relation output by a traitor strategy* $\mathcal{A}$ *based on the user key* α_τ*, such that* $p_\mathcal{D}$ *is non-negligible and* $p_\mathcal{R}$ *is negligible (*cf. *Definition 8). Let* $\hat{\psi}$ *be a* τ*-probe for a message* $\hat{m} \stackrel{R}{\leftarrow} \mathcal{M}$*. Under the* DBDH *assumption,* $\hat{p}_\mathcal{D} \doteq \Pr[\mathcal{R}(\hat{m}, m^*) = 1 \mid m^* \stackrel{R}{\leftarrow} D(\hat{\psi})]$ *is non-negligible.*

Proof. To a contradiction, assume $\hat{p}_\mathcal{D}$ to be negligible. We then construct an efficient algorithm $\mathcal{B}$ that, given pk and the secret key α_τ of a single user, distinguishes valid ciphertexts from τ-probes as follows: on input a ciphertext $\hat{\psi} = \langle \hat{U}, \hat{V}, \hat{W} \rangle$, $\mathcal{B}$ computes $\hat{m} \doteq \hat{W} \oplus H(\hat{U}^{\alpha_\tau} \cdot e(\hat{V}, B_\tau))$ from α_τ and $\hat{\psi}$. Notice that this value $\hat{m}$ is correct regardless of whether $\hat{\psi}$ is a valid ciphertext or a τ-probe. Then, $\mathcal{B}$ feeds $\mathcal{D}$ with $\hat{\psi}$,

getting back a value m^*. If $\mathcal{R}(\hat{m}, m^*) = 1$, then $\mathcal{B}$ concludes that $\hat{\psi}$ must be valid; otherwise, $\mathcal{B}$ concludes that $\hat{\psi}$ is a τ-probe. In other words, $\mathcal{B}$ "interpolates" between the experiment defining probabilities $p_{\mathcal{D}}$ and $\hat{p}_{\mathcal{D}}$, so that $\mathcal{B}$'s advantage in discerning valid ciphertext from τ-probes is clearly $p_{\mathcal{D}} - \hat{p}_{\mathcal{D}}$. But if $\hat{p}_{\mathcal{D}}$ were negligible, such algorithm $\mathcal{B}$ would violate the statement of Lemma 11, proving our argument. □

The next lemma addresses the case of pirate decoders fed with probes of the "wrong type":

Lemma 13. *Replacing $\hat{\psi}$ with a $(1-\tau)$-probe in the setting of Corollary 12,* $\Pr[\mathcal{R}(\hat{m}, m^*) = 1]$ *is negligible.*

Proof. We start with the observation that if we could somehow remove the message $\hat{m}$ from the pirate decoder's view, then our thesis would follow immediately, since $\hat{m}$ would then be independent from the message m^* output by $\mathcal{D}$, and hence, by definition of $p_{\mathcal{R}}$, $\mathcal{R}(\hat{m}, m^*) = 1$ would hold with probability $p_{\mathcal{R}}$, which is negligible.

In fact, $\hat{m}$ occurs in $\mathcal{D}$'s view only in the third component of the $(1-\tau)$-probe $\hat{\psi} \doteq \langle \hat{U}, \hat{V}, \hat{W} \rangle$, as $\hat{W} = \hat{m} \oplus H(\hat{U}^{\alpha_{1-\tau}} e(\hat{V}, B_{1-\tau}))$, so it suffices to show that $\hat{U}^{\alpha_{1-\tau}} e(\hat{V}, B_{1-\tau})$ is indistinguishable from random in $\mathcal{D}$'s view. Since B_0, B_1 both appear in the public key pk of the system, this boils down to proving that $\mathcal{D}$ cannot distinguish $\hat{U}^{\alpha_{1-\tau}}$ from random. It also holds that $\hat{U}^{\alpha_{1-\tau}} = e(P,R)^{k' \alpha_{1-\tau}} = e(P, A_{1-\tau})^{k'}$, so that the task faced by $\mathcal{D}$ is to tell $e(P, A_{1-\tau})^{k'}$ apart from random, given $e(P,R)$, $e(P, A_{1-\tau})$, and $\hat{U} = e(P,R)^{k'}$. But this is just the DDH problem for group $\mathcal{G}_2$, whose hardness is implied by the DBDH assumption.

The above argument can be easily rephrased along the lines of the reductions described in the proofs of Theorem 9 and Lemma 11; we refrain from doing so due to space limitations. □

Theorem 14. *Under the* DBDH *assumption for $(\mathcal{G}_1, \mathcal{G}_2)$, our* **Trace** *algorithm has a negligible traceability error.*

Proof. Let $\mathcal{D}, \mathcal{R}$ be the pirate decoder and resemblance relation on which the **Trace** algorithm is being run, and let τ be the traitor index. Corollary 12 guarantees that **Trace** will on average terminate after $2/p_{\mathcal{D}}$ queries to $\mathcal{D}$. Upon termination, **Trace**'s output will be wrong only if it happens that $\mathcal{D}$ replies to a $(1-\tau)$-probe $\hat{\psi}$ with an m^* satisfying $\mathcal{R}(\hat{m}, m^*) = 1$, *i.e.*, $\Pr[\mathbf{Trace}^{D(\cdot)}(pk, \perp) \notin T] = \Pr[\hat{\psi} \text{ is a } (1-\tau)\text{-probe} \mid \mathcal{R}(\hat{m}, m^*) = 1]$, which by Corollary 12, Lemma 13, and Bayes' theorem is easily seen to equal $p_{\mathcal{R}}/(p_{\mathcal{D}} + p_{\mathcal{R}})$, which is negligible. □

4.4 Our Multi-user Scheme

As mentioned at the beginning of Section 4, a common approach to extending a two-user construction to the multi-user setting is to concatenate several instantiations (say, v) of the basic two-user scheme. Tracing in the resulting multi-user scheme can then be performed iteratively as a sequence of v stages; in each stage, the pirate decoder is queried with ciphertexts that are valid in all v components, except for one, which

instead is crafted according to the **Trace** algorithm of the two-user construction. In this way, if the decoder does not have both sub-keys for the component currently under testing, it will be unable to tell that the ciphertext is invalid, and so the tracing procedure of the two-user subscheme will determine which of the two sub-keys the decoder holds for that component.

Since tracing requires the ability to set up each component of the ciphertext independently of all the others, it may seem necessary to use completely unrelated instantiations of the two-user sub-scheme for each component. This is done, for example, in [16]. Having independent components, however, clearly leads to a multi-user scheme with the same transmission rate as the underlying basic two-user scheme, and so it would not help us attaining optimal transmission rate. In fact, the scheme of [10] manages to get transmission rate 1 by sacrificing component independence, and instead using component-instances all very closely related to each other. As we show in the full version [13], though, their scheme does not support black-box traceability.

To solve this tension between transmission rate and black-box traceability, we move from the observation that, at each stage, it suffices that a single component can be appropriately set up independently from the rest; the remaining $v - 1$ can all be closely related to each other. Therefore, ciphertexts in our construction include a "special" position ℓ, where encryption is performed with instance of our two-user scheme that is specific to the ℓ-th component; the remaining $(v - 1)$ positions, instead, are encrypted using a "shared" two-user scheme.

To prevent pirate decoders from selectively ignoring the "special" position (which is the only part of the ciphertext that encodes tracing information), we follow the approach proposed in [16], by which the encryption algorithm preliminarily processes the plaintext with an *All-Or-Nothing* transform (AONT) [21,8,9]. This will force decoders to decrypt *all* blocks of the ciphertext, since ignoring even a single one would result in missing at least one block of the AONT-transformed plaintext, so that, by the properties of AONT's, such decoders would fail to recover *any* information about the original plaintext being transmitted. We remark that reliance on AONT's to force the pirate to include (at least) one key for each component was suggested in [16], but later dismissed by the authors in [17] as ineffective for the black-box setting, since it cannot prevent cropping of the plaintext once it has been decrypted. However, we believe their critique to be misleading, since traitor strategies in which the pirate decoder tampers with the decrypted plaintexts are dealt with the use of the resemblance relation $\mathcal{R}$ (see discussion in Section 3), while AONT's prevent the pirate from learning anything about the plaintext if even a single block cannot be decrypted.

For the sake of clarity, we first describe the scheme without explicitly mentioning the AONT pre-processing, and later discuss the details regarding the use of AONT's.

Setup: Given the security parameters 1^κ, 1^t and ε, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ of order q, and an admissible bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$. Generate an (ε, t, n, v)-collusion-secure code $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\}$.

Step 2a: Generate v independent copies of the 2-user scheme described in Section 4.1 (call these copies the *special* schemes). In particular, for $j = 1, \ldots, v$, let P_j be a generator of $\mathcal{G}_1$; pick random elements $a_j, b_j, c_j \in \mathbb{Z}_q^*$, and set

$Q_j \doteq a_j P_j$, $R_j \doteq b_j P_j$, $h_j \doteq e(P_j, c_j P_j)$. Also, for $j = 1, \ldots, v$, compute linearly independent vectors $(\alpha_{j,0}, \beta_{j,0})$, $(\alpha_{j,1}, \beta_{j,1}) \in \mathbb{Z}_q^2$ such that $b_j \alpha_{j,\sigma} + a_j \beta_{j,\sigma} = c_j \bmod q$, for $\sigma \in \{0,1\}$.

Step 2b: Generate one more independent copy of the 2-user scheme of Section 4.1, in which we additionally select v values for h (call this the *shared* scheme). At a high level, the shared scheme can be thought of as v parallel copies of the 2-user scheme of Section 4.1, sharing the same values P, Q and R. More precisely, draw $P \xleftarrow{R} \mathcal{G}_1$, $a, b \xleftarrow{R} \mathbb{Z}_q^*$, and set $Q \doteq aP$, and $R \doteq bP$; then, for each $j = 1, \ldots, v$, select $\bar{c}_j \in \mathbb{Z}_q^*$ and set $\bar{h}_j \doteq e(P, \bar{c}_j P)$. Also, for each $j = 1, \ldots, v$, compute two linearly independent vectors $(\bar{\alpha}_{j,0}, \bar{\beta}_{j,0})$, $(\bar{\alpha}_{j,1}, \bar{\beta}_{j,1})$ in $\mathbb{Z}_q^2$ such that $b\bar{\alpha}_{j,\sigma} + a\bar{\beta}_{j,\sigma} = \bar{c}_j \bmod q$, for $\sigma \in \{0,1\}$.

Step 2c: The master secret key msk of the security manager is set to be:

$$((a_j, b_j, (\alpha_{j,0}, \beta_{j,0}, \alpha_{j,1}, \beta_{j,1}))_{j=1,\ldots,v}, a, b, (\bar{\alpha}_{j,0}, \bar{\beta}_{j,0}, \bar{\alpha}_{j,1}, \bar{\beta}_{j,1})_{j=1,\ldots,v})$$

Step 3: For $j = 1, \ldots, v$ and $\sigma \in \{0,1\}$, let $A_{j,\sigma} \doteq \alpha_{j,\sigma} R_j$, $B_{j,\sigma} \doteq \beta_{j,\sigma} P_j$, $\bar{A}_{j,\sigma} \doteq \bar{\alpha}_{j,\sigma} R$ and $\bar{B}_{j,\sigma} \doteq \bar{\beta}_{j,\sigma} P$. Choose a universal hash function $H : \mathcal{G}_2 \to \{0,1\}^\kappa$, and set pk to:[6]

$$(H, (P_j, Q_j, R_j, A_{j,0}, B_{j,0}, A_{j,1}, B_{j,1}), P, Q, R, (\bar{A}_{j,0}, \bar{B}_{j,0}, \bar{B}_{j,1}))$$

for all $j = 1, \ldots, v$. The associated message space is $\mathcal{M} \doteq (\{0,1\}^\kappa)^v$.

Reg: For each user i, the security manager first retrieves the corresponding codeword $\omega_i \in \mathcal{C}$ and sets his/her secret key to: $\mathsf{sk}_i \doteq ((\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v}, (\bar{\alpha}_{j,\omega_j^{(i)}})_{j=1,\ldots,v})$. Notice that, for $j = 1, \ldots, v$, it holds that:

$$c_j P_j = \alpha_{j,\omega_j^{(i)}} R_j + \beta_{j,\omega_j^{(i)}} Q_j \quad \text{and hence} \quad h_j = e(P_j, A_{j,\omega_j^{(i)}}) \cdot e(Q_j, B_{j,\omega_j^{(i)}}),$$
$$\bar{c}_j P = \bar{\alpha}_{j,\omega_j^{(i)}} R + \bar{\beta}_{j,\omega_j^{(i)}} Q \quad \text{and hence} \quad \bar{h}_j = e(P, \bar{A}_{j,\omega_j^{(i)}}) \cdot e(Q, \bar{B}_{j,\omega_j^{(i)}}).$$

Enc: Given pk, anybody can encrypt a message $m = (m^{(1)} \| \ldots \| m^{(v)}) \in \mathcal{M}$ as follows:

First, select $\ell \xleftarrow{R} \{1, \ldots, v\}$ and $k_\ell \xleftarrow{R} \mathbb{Z}_q$, and compute the special component of the ciphertext $(U_\ell, V_\ell, W_\ell) \in \mathcal{G}_2 \times \mathcal{G}_1 \times \{0,1\}^\kappa$, where $U_\ell \doteq e(P_\ell, R_\ell)^{k_\ell}$, $V \doteq k_\ell Q_\ell$ and $W_\ell \doteq m^{(\ell)} \oplus H(h_\ell^{k_\ell})$.

Then, select $k \xleftarrow{R} \mathbb{Z}_q$, and compute the remaining pieces of the ciphertext as: $(U, V, W_1, \ldots, W_{\ell-1}, W_{\ell+1}, \ldots, W_v)$, where $U \doteq e(P, R)^k$, $V \doteq kQ$, and $W_j \doteq m^{(j)} \oplus H(\bar{h}_j^k)$, for $j = 1, \ldots, v$, $j \neq \ell$. The ciphertext is set to be the tuple $\psi \doteq \langle \ell, U_\ell, V_\ell, U, V, W_1, \ldots, W_v \rangle$.

Dec: Given a ciphertext $\psi = \langle \ell, U_\ell, V_\ell, U, V, W_1, \ldots, W_v \rangle \in \mathbb{Z} \times (\mathcal{G}_2 \times \mathcal{G}_1)^2 \times \mathcal{M}$, u_i computes for each $j = 1, \ldots, v, j \neq \ell$:

$$h_\ell^{k_\ell} = (U_\ell)^{\alpha_{\ell,\omega_\ell^{(i)}}} \cdot e(V_\ell, B_{\ell,\omega_\ell^{(i)}}) \quad \text{and} \quad \bar{h}_j^k = (U)^{\bar{\alpha}_{j,\omega_j^{(i)}}} \cdot e(V, \bar{B}_{j,\omega_j^{(i)}})$$

[6] The shared scheme is not used for tracing, so $\bar{A}_{j,1}$ can be safely omitted ($\bar{A}_{j,0}$ is included only so that $\bar{h}_j$ can be computed).

recovers $m^{(\ell)} = W_\ell \oplus H(h_\ell^{k_\ell})$ and $m^{(j)} = W_j \oplus H(\bar{h}_j^{k})$ (for $j \in \{1,\dots,v\} \setminus \{\ell\}$) and outputs $m \doteq (m^{(1)} \| \dots \| m^{(v)})$.

Trace: Given pk, anybody can extract the "traitor codeword" $\hat{\omega} \doteq (\hat{\omega}^{(1)}, \dots, \hat{\omega}^{(v)}) \in \{0,1\}^v$ from a decoder $\mathcal{D}$ by making $O(v)$ queries to $\mathcal{D}$. At a high level, the idea is to iteratively derive each $\hat{\omega}^{(\ell)}$ by feeding $\mathcal{D}$ with an invalid ciphertext that looks valid in the "shared" components, but is actually a *probe* (in the sense of Section 4.3) on the ℓ-th "special" component. In this way, if $\mathcal{D}$ contains only one of the two user-keys for the ℓ-th "special" two-user component (say, $\alpha_{\ell,\tau^{(\ell)}}$), its reply will reveal the value of $\tau^{(\ell)}$. More in detail, to extract $\tau^{(\ell)}$ from $\mathcal{D}$, **Trace** queries $\mathcal{D}$ with ciphertexts of the form $\hat{\psi}^{(\ell)} \doteq \langle \ell, \hat{U}_\ell, \hat{V}_\ell, U^{(\ell)}, V^{(\ell)}, W_1^{(\ell)}, \dots, \hat{W}_\ell^{(\ell)}, \dots, W_v^{(\ell)} \rangle$, where $k_\ell, k'_\ell, k^{(\ell)} \xleftarrow{R} \mathbb{Z}_q$, $\hat{m}^{(\ell)} = \hat{m}_1^{(\ell)}, \dots, \hat{m}_v^{(\ell)}$ is drawn at random from $\mathcal{M}$, $\sigma^{(\ell)}$ is a random bit, $W_j^{(\ell)} \doteq \hat{m}_j^{(\ell)} \oplus H(h_j^{k^{(\ell)}})$ for each $j = 1,\dots,v, j \neq \ell$, and

$$\hat{U}_\ell \doteq e(P_\ell, R_\ell)^{k'_\ell} \qquad \hat{V}_\ell \doteq k_\ell Q_\ell \qquad U^{(\ell)} \doteq e(P,R)^{k^{(\ell)}} \qquad V^{(\ell)} \doteq k^{(\ell)} Q$$
$$\hat{W}_\ell^{(\ell)} \doteq \hat{m}_\ell^{(\ell)} \oplus H(e(P_\ell, A_{\ell,\tau^{(\ell)}})^{k'_\ell} \cdot e(\hat{V}_\ell, B_{\ell,\tau^{(\ell)}})).$$

Let $m^{*(\ell)} \doteq (m_1^{*(\ell)} \| \dots \| m_v^{*(\ell)})$ be the plaintext output by $\mathcal{D}$ when fed with the ciphertext $\hat{\psi}^{(\ell)}$. If $\mathcal{R}(\hat{m}^{(\ell)}, m^{*(\ell)}) = 1$, then set $\hat{\omega}^{(\ell)} = \sigma^{(\ell)}$; otherwise, pick fresh random $k_\ell, k'_\ell, k^{(\ell)}$ from $\mathbb{Z}_q$, $\hat{m}^{(\ell)}$ from $\mathcal{M}$, $\sigma^{(\ell)}$ from $\{0,1\}$, and repeat, until either $\mathcal{R}(\hat{m}^{(\ell)}, m^{*(\ell)}) = 1$, or the iteration has failed some fixed polynomial number of time, in which case $\hat{\omega}^{(\ell)}$ is set arbitrarily.

After this process has been repeated for $\ell = 1,\dots,v$, the resulting "traitor codeword" $\hat{\omega}$ is handed to the tracer, who (knowing the random coins $r_{\mathcal{C}}$ used in generating $\mathcal{C}$) can run it through the tracing algorithm $\mathcal{T}(r_{\mathcal{C}}, \cdot)$ of the collusion-secure code $\mathcal{C}$, thus obtaining a value in $\{1,\dots,n,0\}$, which is the output of **Trace**.

Remark 15. Since the **Trace** algorithm needs msk only in the off-line phase, which does not access the pirate decoder and is much less computation-intensive,[7] our multi-user scheme supports local public traceability.

Remark 16. We bound the number of trials that **Trace** performs to extract each bit $\hat{\omega}^{(\ell)}$ because a pirate decoder holding both keys for position ℓ could cause the test $\mathcal{R}(\hat{m}^{(\ell)}, m^{*(\ell)}) = 1$ to fail with probability 1. A suitable value for this bound is $O(1/p_{\mathcal{D}})$, where $p_{\mathcal{D}}$ is the success probability (over random valid ciphertexts) of the decoder under tracing, which can be efficiently estimated using Chernoff bounds.

Remark 17. Notice that the size of the message blocks can be shrunk to any $\kappa' \leq \kappa$, by choosing a universal hash function $H : \mathcal{G}_2 \to \{0,1\}^{\kappa'}$. This is possible as long as $\kappa' > \log v + \log(1/\varepsilon) = O(\log t + \log\log(n/\varepsilon) + \log(1/\varepsilon))$, which ensures that, during tracing, the probability of a hash collision in any of the v components of the scheme is bounded by ε. For a typical choice of parameters ($n = 2^{30}$, $\varepsilon = 2^{-30}$, $t = 30$), κ' can be chosen as low as 64 bits.

[7] For the scheme of [23], for example, such computation consists just of a matrix-vector multiplication.

Pre-Processing Messages with AONT's. An AONT is an efficient, unkeyed, randomized transformation, with the property that it is hard to invert unless the *entire* output is known. (For a formal definition, see [8,9].) As for specific instantiations, Boyko showed in [8] that the *Optimal Asymmetric Encryption Padding* (OAEP)[3] can be proven secure as an AONT in the Random Oracle Model. In [9], Canetti *et al.* described constructions in the standard model based on the notion of *Exposure-Resilient Functions*.

For our purposes, it suffices to think of an AONT as a length-preserving algorithm $\mathrm{AONT}(m;r)$, where $m \in (\{0,1\}^\kappa)^{v-1}$ is the message to be processed and r is an additional random value, of the same length as each message block *i.e.*, $|r| = \kappa$. In what follows, we denote by $M \stackrel{R}{\leftarrow} \mathrm{AONT}(m)$ the process of selecting a random r from $\{0,1\}^\kappa$ and setting $M \leftarrow \mathrm{AONT}(m;r)$. The resulting AONT-transformed message $M = (M_1, \ldots, M_v)$ is an element of $(\{0,1\}^\kappa)^v$, so that it can be encrypted with the $\mathbf{Enc}$ algorithm described above. We can thus define a multi-user scheme with AONT pre-processing by modifying the $\mathbf{Enc}$ and $\mathbf{Dec}$ algorithms as:

$$\mathbf{Enc}'(m) \doteq \mathbf{Enc}(\mathrm{AONT}(m)) \qquad \mathbf{Dec}'(\psi) \doteq \mathrm{AONT}^{-1}(\mathbf{Dec}(\psi))$$

Notice that the use of AONT pre-processing in the full-blown scheme implies an expansion in the message size by roughly a factor $1 + 1/v$, which still results in an asymptotical unitary ciphertext-to-plaintext ratio.

4.5 Indistinguishability Under Chosen-Plaintext Attack

In this section, we assess the security of the multi-user scheme of Section 4.4. (For lack of space, we defer all proofs for this section to the full version [13]).

We start by verifying the intuition that AONT pre-processing does not hurt security:

Lemma 18. *If the multi-user scheme without AONT pre-processing is secure w.r.t. indistinguishability under chosen-plaintext attack, then the multi-user scheme with AONT pre-processing is secure w.r.t. the same notion.*

Next, we observe that the security of the multi-user scheme from Section 4.4 can be reduced (via a hybrid argument) to the security of the two-user scheme from Section 4.1:

Lemma 19. *If our two-user scheme is secure w.r.t. indistinguishability under chosen-plaintext attack, then our multi-user scheme without AONT pre-processing is secure w.r.t. the same notion.*

In light of Theorem 9, our main security theorem follows immediately from Lemmata 18 and 19:

Theorem 20. *Under the* DBDH *assumption for* $(\mathcal{G}_1, \mathcal{G}_2)$*, the scheme in Section 4.4 is secure w.r.t. indistinguishability under chosen-plaintext attack.*

4.6 Traceability

Similarly to the case of the 2-user scheme of Section 4.1, the traceability of our multi-user scheme (with AONT pre-processing) is based on the notions of *valid* and *probe* ciphertexts:

Definition 21. *Let* $\ell \in [1,v]$, $\sigma \in \{0,1\}$, $\hat{m} \in \mathcal{M}$, $\hat{M} = (\hat{M}_1,\ldots,\hat{M}_v) \xleftarrow{R} \mathrm{AONT}(\hat{m})$, $\hat{U}_\ell \in \mathcal{G}_2$, $\hat{V}_\ell \in \mathcal{G}_1$, $k \in \mathbb{Z}_q$, $U = e(P,R)^k$, $V = kQ$, $W_j = \hat{M}_j \oplus H(h_j^k)$ $(j = 1,\ldots,v,\ j \neq \ell)$, $\hat{W}_\ell = \hat{M}_\ell \oplus H(\hat{U}_\ell^{\alpha_{\ell,\sigma}} e(\hat{V}_\ell, B_{\ell,\sigma}))$, *and* $\hat{\psi} = \langle \ell, \hat{U}_\ell, \hat{V}_\ell, U, V, W_1, \ldots, \hat{W}_\ell, \ldots, W_v \rangle$. *We say that the ciphertext* $\hat{\psi}$ *is:*

- ***valid****, if* $\hat{U}_\ell = e(P_\ell, R_\ell)^{k_\ell}$, $\hat{V}_\ell = k_\ell Q_\ell$, *for some* $k_\ell \in \mathbb{Z}_q$;
- (ℓ,σ)***-probe****, if* $\hat{U}_\ell = e(P_\ell, R_\ell)^{k'_\ell}$, $\hat{V}_\ell = k_\ell Q_\ell$, *for distinct* $k_\ell, k'_\ell \in \mathbb{Z}_q$.

Our analysis is organized as follows. Let T denote the set of indices of the t traitors. Lemma 22 proves the computational indistinguishability of valid ciphertexts *vs.* (ℓ,τ^ℓ)-probes when only the τ^ℓ subkey is known for position ℓ. It follows (Corollary 23) that pirate decoders must decrypt such (ℓ,τ^ℓ)-probes correctly (w.r.t. the chosen resemblance relation). Lemma 24 then shows that instead $(\ell, 1-\tau^\ell)$-probes cannot be properly decrypted, and Lemma 25 combines Corollary 23 and Lemma 24 to argue that the chances that the ℓ-th stage of tracing fails to extract the correct bit $\hat{\omega}^{(\ell)} = \tau^\ell$ from $\mathcal{D}$ are negligible, which implies the overall traceability of our scheme (Theorem 26).

Lemma 22 (Indistinguishability of Valid *vs.* Probe Ciphertexts). *Under the* DBDH *assumption for* $(\mathcal{G}_1,\mathcal{G}_2)$, *given the public key* $\mathsf{pk} = (q, \mathcal{G}_1, \mathcal{G}_2, e, H, P_j, Q_j, R_j, (A_{j,0}, B_{j,0}, A_{j,1}, B_{j,1})_{j=1,\ldots,v}, P, Q, R, (\bar{A}_{j,0}, \bar{B}_{j,0}, \bar{B}_{j,1})_{j=1,\ldots,v})$ *and the secret keys* $\mathsf{sk}_i \doteq ((\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v}, (\bar{\alpha}_{j,\omega_j^{(i)}})_{j=1,\ldots,v})$ *for each* $i \in T$, *it is infeasible to distinguish valid ciphertexts from* (ℓ,τ^ℓ)*-probes, if the codewords of all traitors in* T *have bit* τ^ℓ *at position* ℓ.

Proof. Since the ℓ-th "special" sub-schemes is completely independent from the rest of our construction, the thesis follows as a simple reduction to Lemma 11. □

Corollary 23. *Let* $\mathcal{D}$, $\mathcal{R}$ *be the pirate decoder and resemblance relation output by a traitor strategy* $\mathcal{A}$ *based on the user keys of the traitors in* T, *such that* $p_{\mathcal{D}}$ *is non-negligible and* $p_{\mathcal{R}}$ *is negligible (*cf. *Definition 8). Assume the codewords of all the traitors in* T *have bit* τ^ℓ *at position* ℓ, *and let* $\hat{\psi}$ *be an* (ℓ,τ^ℓ)*-probe for a message* $\hat{m} \xleftarrow{R} \mathcal{M}$. *Under the* DBDH *assumption,* $\hat{p}_{\mathcal{D}} \doteq \Pr[\mathcal{R}(\hat{m}, m^*) = 1 \mid m^* \xleftarrow{R} D(\hat{\psi})]$ *is non-negligible.*

Proof. Reduces to Lemma 22 exactly as Corollary 12 reduces to Lemma 11.

Lemma 24. *Replacing* $\hat{\psi}$ *with an* $(\ell, 1-\tau^\ell)$*-probe in the setting of Corollary 23,* $\Pr[\mathcal{R}(\hat{m}, m^*) = 1]$ *is negligible, if the AONT employed in the system is secure.*

Proof. The argument described in the proof of Lemma 13 implies that the AONT-transformed message block $\hat{M}_\ell$ is computationally hidden from the pirate decoder's view. By the properties of AONT's, the whole original message $\hat{m}$ is then also computationally hidden from $\mathcal{D}$, so that in fact $\hat{m}$ is just a random message independent from the output m^* of $\mathcal{D}$, and hence $\mathcal{R}(\hat{m}, m^*) = 1$ holds with probability $p_{\mathcal{R}}$, which is negligible. □

Lemma 25. *Consider the* ℓ*-th stage of the* **Trace** *algorithm, when the tracer queries the decoder* $\mathcal{D}$ *with* (ℓ,σ)*-probes for random* $\sigma \in \{0,1\}$. *If all codewords of the traitors in* T *have bit* τ^ℓ *in the* ℓ*-th position, then the* ℓ*-th stage will terminate setting* $\hat{\omega}^\ell = 1-\tau^\ell$ *with negligible probability.*

Proof. The assumption that $\mathcal{D}$ does not contain both keys for position ℓ implies, by Corollary 23, that the ℓ-th stage of **Trace** will on average terminate after $2/p_{\mathcal{D}}$ queries to $\mathcal{D}$. Upon termination, **Trace**'s output will be wrong only if it happens that $\mathcal{D}$ replies to an $(\ell, 1-\tau^{\ell})$-probe $\hat{\psi}$ with an m^* satisfying $\mathcal{R}(\hat{m}, m^*) = 1$, which by Corollary 23, Lemma 24, and Bayes' theorem is easily seen to equal $p_{\mathcal{R}}/(p_{\mathcal{D}} + p_{\mathcal{R}})$, which is negligible. □

Theorem 26. *Under the* DBDH *assumption for* $(\mathcal{G}_1, \mathcal{G}_2)$*, the multi-user* **Trace** *algorithm from Section 4.4 has a negligible traceability error.*

Proof. Let $\hat{\omega} = (\hat{\omega}^{(1)}, \ldots, \hat{\omega}^{(v)})$ be the "traitor codeword" recovered at the end of the publicly traceable phase of **Trace** (*cf.* Section 4.4). By the union bound, Lemma 25 implies that $\hat{\omega}$ will be correct in all positions ℓ where all traitors show the same bit, except with negligible probability. By the collusion resistance of the code $\mathcal{C}$ underlying the key assignment of **Setup**, the codeword-tracing algorithm $\mathcal{T}$ (*cf.* Definition 4) will then be able to tie such traitor codeword $\hat{\omega}$ to the identity of one of the traitors in T (except with negligible probability ε), as required. □

Remark 27. As noted above, by employing AONT's, the security and tracing capabilities of our multi-user scheme follow almost directly from those of the embedded "special" sub-scheme. In fact, even if we were to suppress the shared sub-scheme (*e.g.*, by setting $W_j = M_j$, for $j = 1, \ldots, v, j \neq \ell$), the multi-user scheme would still be secure and tracing would still be possible (thanks also to the random rotation of the special position ℓ between 1 and v). Using the shared sub-scheme, however, reinforces the semantic security of the scheme, though at the cost of a greater computational load, due to the larger number of pairing computations needed for encryption and decryption.

5 Space and Time Parameters in a Concrete Instantiation

Existing constructions of constant-rate traitor tracing schemes (including ours) are based on the use of collusion-secure fingerprint codes[8] [6,23], and in particular are applicable for messages of size proportional to the length of the code, which in the case of the optimal codes due to Tardos [23] is $O(t^2(\log n + \log \frac{1}{\varepsilon}))$. For a typical choice of parameters, *e.g.* user population $n = 2^{30}$, tracing error probability $\varepsilon = 2^{-30}$ and traceable threshold $t = 30$, the resulting code length is about 5 million bits. Instantiating our construction with such codes yields a scheme with plaintext and ciphertext of size 41 MBytes. (The ciphertext size is equal to the plaintext size, as the additive overhead is less than 1 KByte.) These values are well within the range of multimedia applications, since 41 MBytes roughly corresponds to 33 seconds of DVD-quality (high-resolution) video, 4 minutes of VCD-quality (low-resolution) video and 25–50 minutes of audio. The resulting public and secret keys roughly require respectively 1.5GByte and 206 MBytes. Although quite large, such a public key could be stored in commodity hardware (*e.g.,* it would fit in the hard disk of an iPod), whereas user secret keys could be kept in Secure Digital memory cards, like those commonly available for PDAs.

[8] [20] actually employs IPP codes, but similar considerations on code length and message size apply to such codes as well.

Another important issue for a concrete instantiation is the rate at which encrypted content can be processed. In our scheme, decryption requires one paring per 1024 bits of content, which, using the PBC Library [18] on a desktop PC, takes approximately 16 msec. However, in our context, the pairings to be computed all have one of their two input-points in common: as reported in [2], pre-processing in similar settings more than halves the computation time, so that one easily gets in the order of 128 pairings/sec, corresponding to a near-CD-quality audio rate of 128 Kbits/sec. More specialized software implementations [1] of the pairing operation can further reduce its computational cost to around 3 msec; whereas hardware implementations, even under conservative assumptions on the hardware architecture [14], can obtain running time below 1 msec, attaining the 1Mbits/sec data rate needed for VCD-quality video.

6 Conclusion

We present the first public-key traitor tracing scheme with efficient black-box tracing and optimal transmission rate. Our treatment improves the standard modeling of black-box tracing by additionally accounting for pirate strategies that attempt to escape tracing by purposedly rendering the transmitted content at lower quality (*e.g.* by dropping every other frame from the decrypted video-clip, or skipping few seconds from the original audio file). We also point out and resolve an issue in the black-box traitor tracing mechanism of both the previous schemes in this setting [16,10]. Our construction is based on the decisional bilinear Diffie-Hellman assumption, and additionally provides the same features of public traceability as (a repaired version of) [10], which is less efficient and requires non-standard assumptions for bilinear groups.

References

1. Barreto, P., Galbraith, S., hEigeartaigh, C., Scott, M.: Efficient Pairing Computation on Supersingular Abelian Varieties (2004), Available at http://eprint.iacr.org/2004/375
2. Barreto, S.L.M., Ben, L., Scott, M.: Efficient Implementation of Paring-Based Cryptosystems. Journal of Cryptology 17(4), 321–334 (2004)
3. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption–How to Encrypt with RSA (LNCS 950). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
4. Boneh, D., Franklin, M.: An Efficient Public Key Traitor Tracing Scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)
5. Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
6. Boneh, D., Shaw, J.: Collusion-Secure Fingerprinting for Digital Data. IEEE Transactions on Information Theory 44(5), 1897–1905 (1998)
7. Boneh, D., Waters, B.: A Collusion Resistant Broadcast, Trace and Revoke System. In: Computer and Communication Security—CCS'06, pp. 211–220. ACM Press, New York (2006)
8. Boyko, V.: On the Security Proprties of the OAEP as an All-or-Nothing Transform. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 503–518. Springer, Heidelberg (1999)

9. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-Resilient Functions and All-or-Nothing Transform. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
10. Chabanne, H., Phan, D.H., Poitcheval, D.: Public Traceability in Traitor Tracing Schemes. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 542–558. Springer, Heidelberg (2005)
11. Chor, B., Fiat, A., Naor, N.: Tracing Traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
12. Chor, B., Fiat, A., Naor, N., Pinkas, B.: Tracing Traitors. IEEE Transaction on Information Theory 46(3), 893–910 (2000)
13. Fazio, N., Nicolosi, A., Phan, D.H.: Traitor tracing with optimal transmission rate. In: Information Security Conference—ISC'07, Full version (2007), available at http://cs.nyu.edu/~fazio/research/research.html
14. Kerins, T., Marnane, W.P., Popovici, E.M., Barreto, P.S.L.M.: Efficient Hardware for the Tate Paring Calculation in Characteristic Three. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 412–426. Springer, Heidelberg (2005)
15. Kiayias, A., Yung, M.: Self Protecting Pirates and Black-Box Traitor Tracing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 63–79. Springer, Heidelberg (2001)
16. Kiayias, A., Yung, M.: Traitor Tracing with Constant Transmission Rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)
17. Kiayias, A., Yung, M.: Copyrighting Public-key Functions and Applications to Black-box Traitor Tracing. Full revised version of [16], Available at (2006), http://eprint.iacr.org/2006/458/
18. Lynn, B.: PBC Library. Available at, http://crypto.stanford.edu/pbc/
19. Naor, M., Pinkas, B.: Threshold Traitor Tracing. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 502–517. Springer, Heidelberg (1998)
20. Phan, D.H., Phan, D.H., Safavi-Naini, R., Tonien, D.: Generic construction of hybrid public key traitor tracing with full-public-traceability. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 264–275. Springer, Heidelberg (2006)
21. Rivest, R.: All-or-Nothing Encryption and the Package Transform. In: Fast Software Encryption (1997)
22. Stinson, D.R., Wei, R.: Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. SIAM Journal on Discrete Mathematics 11(1), 41–53 (1998)
23. Tardos, G.: Optimal Probabilistic Fingerprint Codes. In: Proceedings of the 35th Symposium on Theory of Computing—STOC'03, pp. 116–125. ACM Press, New York (2003)

The Security of Elastic Block Ciphers Against Key-Recovery Attacks

Debra L. Cook[1,⋆], Moti Yung[2], and Angelos D. Keromytis[2]

[1] dcook@cs.columbia.edu
[2] Department of Computer Science, Columbia University, New York, NY, USA
{moti,angelos}@cs.columbia.edu

Abstract. We analyze the security of elastic block ciphers against key-recovery attacks. An elastic version of a fixed-length block cipher is a variable-length block cipher that supports any block size in the range of one to two times the length of the original block. Our method for creating an elastic block cipher involves inserting the round function of the original cipher into a substitution-permutation network. In this paper, we form a polynomial-time reduction between the elastic and original versions of the cipher by exploiting the underlying network structure. We prove that the elastic version of a cipher is secure against a given key-recovery attack if the original cipher is secure against such an attack. Our analysis is based on the general structure of elastic block ciphers (*i.e.,* the network's structure, the composition methods between rounds in the network and the keying methodology) and is independent of the specific cipher.

Keywords: Variable-length block ciphers, security analysis, reduction proof, key recovery attacks.

1 Introduction

Elastic block ciphers are variable-length block ciphers created from existing block ciphers [5]. The elastic version of a block cipher supports any block size between one and two times that of the original block length, and results in a computational workload for encryption that is proportional to the actual block size. Our method for creating elastic block ciphers consists of a substitution-permutation network that uses the round function from the existing fixed-length block cipher as a black box. Elastic block ciphers, in turn, can be combined with modes of encryption to support encryption of any size cleartext.

Traditionally, block ciphers are designed to support a specific block size, with the security analysis and design optimized for the supported block size. For a variable-length block cipher, a more general analysis is required to avoid evaluating the cipher separately for each supported block length. Furthermore, for elastic block ciphers it is preferable to be able to analyze the ciphers as a category as opposed to evaluating each one individually against specific attacks to which the fixed-length versions have previously been proven to be immune.

⋆ This work was performed while the author was at Columbia University.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 89–103, 2007.

We have extensively analyzed both the underlying structure used to create elastic block ciphers and practical examples of elastic block ciphers. Our analysis has ranged from proving that elastic block ciphers, in theory, provide variable length pseudorandom permutations (PRPs) and strong PRPs to creating and analyzing concrete examples [4]. In this work, we present our analysis of the security of elastic block ciphers against practical attacks. These attacks typically attempt to recover the keys or the round keys of the block cipher. Differential cryptanalysis [3,7], linear cryptanalysis [9] and exhaustive search methods are instances of such attacks (but other key-recovery attacks exist [2,13]).

We prove, in general, that the elastic version of a block cipher is secure against attacks that attempt to recover key bits if the original, fixed-length version of the cipher is secure against such attacks. *Our method is unique in that we show how to convert such an attack on the elastic version directly into an attack on the original version with a polynomially related time complexity.* Unlike generic design methodologies, where the component from which security is derived is a well defined black-box building block [6], our proof requires identifying the presence of a fixed-length instance of the block cipher embedded inside the elastic design even though it is the round function and not the original block cipher in its entirety that is used as a black box. As a result of our proof, if the original cipher is (assumed, shown heuristically, or proven to be) immune to a certain type of attack (such as linear or differential cryptanalysis) then the elastic version is also (respectively assumed, shown heuristically, or proven to be) immune to the attack in the same sense (with polynomially related parameters that we concretely calculate).

The use of the round function of the original block cipher as a black box in the elastic version, together with the methods by which we compose rounds and schedule key material, is what enables us to relate the security of the elastic version of a block cipher directly to the security of the original cipher. Our general approach is motivated by reduction-oriented proofs of security. Such proof techniques are not typical in symmetric-key cryptography, especially in concrete designs (for a survey of proof techniques in this area, see [12]:Chapter 4), and are more common in generic designs that assume strong secure components (*e.g.,* assuming a component is a random function or a pseudorandom function [8]).

Our elastic block cipher design exploits existing components of a cipher to gain efficiency and avoids using the entire fixed-length cipher as a black-box (as was done in earlier work, [1,11]). Thus, it may appear at first that the ability to perform a reduction-based proof is lost. However, the methodology presented in this work demonstrates that even concrete designs that use components of a cipher may resort to reduction-like proof techniques if the components' properties and the composition methods are carefully chosen, even with respect to concrete key-recovery attacks as opposed to only distinguishability attacks, which are more typical in investigations of a formal theoretical nature. To the best of our knowledge, this type of methodology is new in this area. While it is not common in block cipher design, we believe it will be a useful analysis tool in settings that employ cipher components within extended contexts, and may also be of independent interest.

The remainder of the paper is organized as follows. Section 2 summarizes the construction of elastic block ciphers. Section 3 defines the relationship between the security of the elastic version of a block cipher against key recovery attacks to the security of the original cipher against such attacks. Section 4 concludes the paper.

2 Elastic Block Cipher Review

2.1 Overview

We briefly review our method for creating elastic block ciphers [5]. The method converts the encryption and decryption functions of any existing block cipher to accept blocks of size b to $2b$ bits, where b is the block size of the original block cipher. The general structure of an elastic block cipher is shown in Figure 1. An elastic version of a block cipher is created by inserting the cycle of the original fixed-length block cipher into the network structure to form the round function of the elastic version. In each round the leftmost b bits are processed by the round function and the rightmost y bits are omitted from the round function. Afterwards, the rightmost y bits are XORed with a subset of the leftmost b bits and the results swapped. This swapping of bits may be omitted after the last round. The number of rounds in the elastic version is set such that the round function is applied to each bit position at least the same number of times as in the fixed-length version. The elastic version also includes initial and end-of-round whitening, and an initial and final key-dependent permutation. The key-dependent permutations are present to prevent an attacker from knowing with a probability of 1 exactly what y bits are omitted from the first application of the round function when encrypting or decrypting. Decryption is performed by applying the network in reverse with the round function of G' replaced by its inverse, specifically the inverse of the cycle in G.

We use the following notation from the definition of elastic block ciphers [5] throughout the remainder of this paper.
Notation:

- G denotes any existing block cipher with a fixed-length block size that is structured as a sequence of rounds. By default, any block cipher that is not structured as a sequence of rounds is viewed as having a single round.
- A cycle in G refers to the point at which all b-bits of the block have been processed by the round function of G. For example, if G is a Feistel network, a cycle is the sequence of applying the round function of G to the left and right halves of the b-bit block. In AES [10], the round function is a cycle.
- r denotes the number of cycles in G.
- b denotes the block length of the input to G in bits.
- y is an integer in the range $[0, b]$.
- G' denotes the modified G with a $(b + y)$-bit input for any valid value of y. G' will be referred to as the elastic version of G.
- r' denotes the number of rounds in G'.
- The round function of G' will refer to one entire cycle of G.
- The swap step will refer the step in which the rightmost y bits are XORed with y bits from the leftmost b bits and the results swapped.

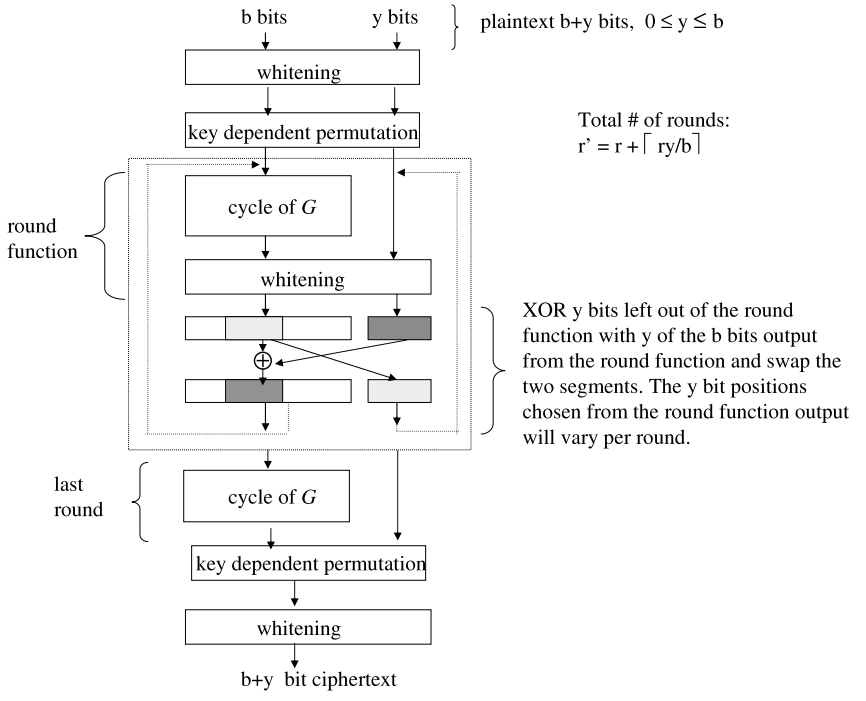


Fig. 1. Elastic Block Cipher Structure

The elastic version of a block cipher requires a greater number of expanded key bits than the original, fixed-length version. In practice, options for the key schedule include using a stream cipher to generate all expanded key bits, applying the original key schedule multiple times, or using the original key schedule for some expanded key bits and a stream cipher or other algorithm for the additional key bits. We note that the use of a stream cipher for the key schedule allows for a generic key schedule across all elastic block ciphers and increases the pseudorandomness of the expanded key bits when compared to existing key schedules, although in practice this incurs the cost of a decrease in the rate of key expansion [4]. The acceptable relationships between the expanded key bits of the elastic version and the original key bits are expressed in the security analysis below.

3 Security Analysis

3.1 Overview

For any concrete block cipher used in practice, as opposed to a theoretical construction of a pseudorandom permutation (PRP), the cipher cannot be proven secure in a formal sense (is not proven to be a PRP or strong PRP) but rather is proven or shown under

certain assumptions to be secure against known types of attacks. Thus, we can only do the same for the elastic version of such a cipher. In order to provide a general understanding of the security of elastic block ciphers, we provide a method for reducing the security of the elastic version to that of the original version, showing that a security weakness in G' implies a weakness in G. Our security analysis of G' exploits the fact that there is an instance of G embedded in G' and is independent of the specific block cipher used for G.

We prove that G' is secure against any attack that attempts to recover the key or the expanded-key bits if G is secure against the attack, under certain assumptions on the independence of the expanded-key bits in G'. This is accomplished by showing how to convert such an attack on G' to an attack on G. We believe this result is important because it implies that G' does not have to be analyzed against any practical attack to which G is immune (unless a more refined analysis than the reduction is required). Our approach is novel because we show how to convert an attack on the variable-length version of a block cipher directly into an attack on the fixed-length version of the block cipher, and, in general, it points out at a direction of identifying embedded ciphers inside ciphers when the design is not purely of a black box fashion.

Security against key recovery attacks does not by itself imply security (*e.g.,* the identity function which ignores the key is insecure while key recovery is impossible). However, all concrete attacks against real ciphers (linear, differential, higher order differential, impossible differential, related key attacks, *etc.*) attempt key or expanded-key recovery and thus practical block ciphers should be secure against such attacks. We note that if there is a relationship between the plaintext and ciphertext bits that does not involve the key bits, this relationship would either manifest itself in the results of statistical tests on whatever versions of the block cipher (original and/or elastic) for which the relationship holds, and/or as algebraic equations relating the plaintext and the ciphertext.

3.2 G Within G'

Before stating our theorem, we provide some preliminary analysis that assists us in conveying the linkage between the original and elastic versions of a block cipher. For simplification of terminology only, we will refer to the fixed-length block cipher G as if the round function of G is a cycle and omit using the term "cycle". For any G in which a cycle involves multiple applications of the round function, such as in a Feistel network, our analysis holds by referring to a cycle of G instead of the round function of G.

We first draw attention to the fact that the operations performed in G' on the leftmost b-bit positions in r consecutive rounds is an application of G. This is depicted intuitively in Figure 2. We note that we are concerned only with r consecutive rounds of G' and do not include either the initial or final key-dependent permutation present in the definition of elastic block ciphers. This relationship between G' and G can be used to convert an attack which finds the round keys for G' to an attack which finds the round keys for G. Let G_{rk} denote G using round keys rk and let ${G_k}'$ denote G' using key k. Let (p, c) be a b-bit (plaintext, ciphertext) pair, and let x and z each be of length y. $\|$ denotes concatenation. If ${G_k}'(p \parallel x) = c \parallel z$, a set of round keys, rk, for G such that $G_{rk}(p) = c$ can be formed from the round keys and the round outputs in G' by

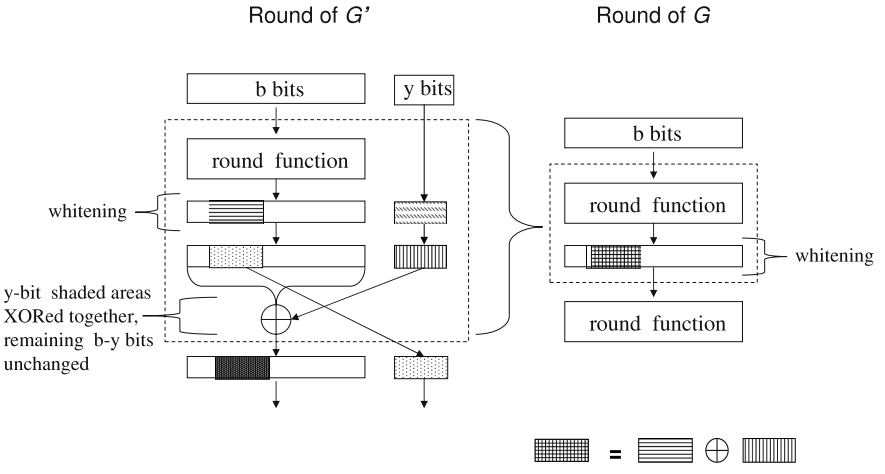


Fig. 2. G within G'

collapsing the end-of-round whitening and swap steps in G' into a whitening step. The leftmost b bits of the initial whitening in G' are used as the initial whitening in G and the rightmost y bits of the initial whitening in G' are dropped. The resulting end-of-round whitening key bits for G will vary in up to y positions across the (plaintext, ciphertext) pairs when collapsing the steps from G'; however, it is possible to use these keys to solve for the round keys of G.

The following claim shows that for any set of (plaintext, ciphertext) pairs encrypted under sets of round keys in G' where the rightmost y bits used for whitening in each round may vary amongst the sets and all other key bits are identical amongst the sets, there exists a corresponding set of (plaintext, ciphertext) pairs for G where the round keys used in G' for the round function and the leftmost b bits of each whitening step are the same as those used in G, the plaintexts used in G are the leftmost b bits of the plaintexts used in G', and the ciphertexts for G are the leftmost b bits of output of the r^{th} round of G' prior to the swap step.

Claim 1: Let G be a b-bit block cipher and G' be its elastic version. Let $\{(pi, ci)\}$ denote a set of n (plaintext, ciphertext) pairs such that $|pi| = |ci| = b$. Let $b + y$ be the variable block size for G' where $0 \leq y \leq b$. Let w be a y-bit constant. Let vi be a y bit string that may vary per i, for $i = 1$ to n. Under the following assumptions regarding the key schedules:

- The rightmost y bits of each whitening step in G' can take on any value and are independent of any other expanded-key bits within the round and in other rounds.
- There are no message-related expanded keys. Any expanded-key bits utilized in G depend only on the key and do not vary across plaintext or ciphertext inputs.
- Any expanded-key bits used in the round function of the r consecutive rounds of G' can take on the same values as the expanded-key bits used in the round functions of G.

– If G contains initial and end-of-round whitening, any expanded-key bits used for the leftmost b bits of each whitening step in r consecutive rounds of G' can take on the same values as the whitening bits in G.

if $G_k(pi) = ci$ then there exists n sets of round keys for the first r rounds of G' that are consistent with inputs $pi \parallel w$ producing $ci \parallel vi$ as the output of the r^{th} round prior to the swap step at the end of the r^{th} round, for $i = 1$ to n, such that the leftmost b bits used for whitening in each round are identical across the n sets and any expanded-key bits used internal to the round function are identical across the n sets.

Proof. Let $rk = \{rk_0, rk_1, ...rk_r\}$ be the set of round keys corresponding to key k for G. rk_0 denotes the key bits used for initial whitening. For each (pi, ci), form a set of the first r round keys for G' as follows: Pick a constant string, w, of y bits, such as a string of $0's$. Let $pi \parallel w$ be the input to G'. Let $rki' = \{rki'_0, rki'_1, ...rki'_r\}$ denote the round keys for G' through the r^{th} round for the pair (pi, ci). Set any bits in rki'_j used internal to the round function to be the same as the corresponding bits in rk_j. Set the leftmost b bits used for whitening in rki'_j to the b bits used for whitening in rk_j. Set the rightmost y bits used for whitening in rki'_j to be the same as the y bits left out of the round function in round j of G'. This is illustrated in Figure 3. Notice that the leftmost b bits used for whitening in each round are identical across the n sets of round keys formed, and any bits used internal to the round function are identical across the n sets; specifically, they correspond to rk in each case, and the rightmost y bits used in each whitening step differ based on (pi, ci) across the n sets. The case in which G does not contain whitening steps corresponds to using 0's for the leftmost b bits of each whitening step in G'.

The operations of G' on the leftmost b bits of rounds 1 through round r, prior to the last swap, are identical to the operations in $G_k(pi)$ because the swap step in G' results

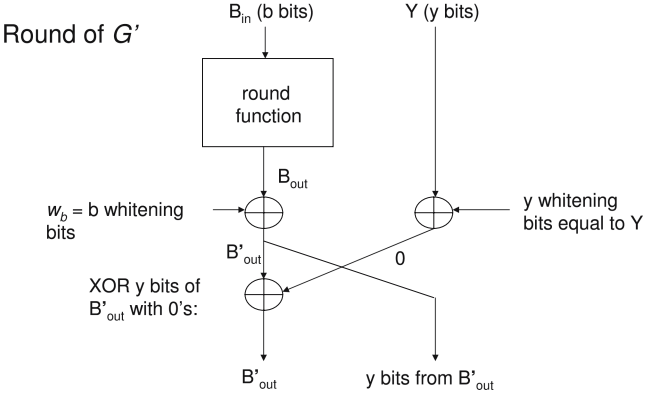


Fig. 3. Converted Key Unchanged in b Whitening Bits

in XORing y bits of a round function's output with y $0's$. Thus, the leftmost b bits in the output of the r^{th} round prior to the swap step is ci. Therefore, for $i = 1$ to n there exists a set of round keys, rki' for $G'_{rki'}$ such that $G'(pi)$ produces ci as the leftmost b bits in the r^{th} round prior to the swap step, thus proving the claim.

3.3 Reduction Between the Original and Elastic Versions of a Cipher

We use the fact that an instance of G is embedded in G' to create a reduction from G' to G. As a result of this reduction, an attack against G' that allows an attacker to determine some of the round keys implies an attack against G that is polynomially related in resources to the attack on G'. Assuming that G itself is resistant to such attacks, we conclude that G' is also resistant to such attacks. We note that if an attack finds the key as opposed to the expanded-key bits (the round keys) then the attacker can apply the key schedule to the key to obtain the round keys. Therefore, in our analysis, we view any key recovery attack as providing the round keys to the attacker. The reduction requires a set of (plaintext, ciphertext) pairs. This is not considered a limiting factor because in most types of attacks, whether they are known plaintext, chosen plaintext, adaptive chosen plaintext, chosen ciphertext *etc.,* the attacker acquires a set of such pairs.

In our analysis, we consider G' without the initial and final key-dependent permutations. This allows us to focus on the core components of the elastic block cipher algorithm. If present, the initial and final permutations only serve to increase the security of G' since they prevent an attacker from knowing with probability one which bits are omitted from the first application of the round function when encrypting or decrypting. Furthermore, since these permutations are added steps (as opposed to modifications to components of G) using key material that is independent of the round and whitening key bits, they do not impact our analysis.

Theorem 1. *Given a fixed-length block cipher, G, that works on b-bit blocks and its elastic version, G', that works on $(b+y)$-bit blocks, where $0 \leq y \leq b$, if there exists an attack, $A'_{G'}$, on G' that allows the round keys to be determined for r consecutive rounds of G' using polynomial (in b and/or r) time and memory, then there exists an attack on G with r rounds that finds the round keys for G and that uses polynomial (in b and/or r) many resources as $A'_{G'}$, assuming:*

- *There are no message-related expanded keys. Any expanded-key bits utilized in G depend only on the key and do not vary across plaintext or ciphertext inputs.*
- *An attack on r' rounds of G' implies a reduced-round attack on r rounds of G' for $r \leq r'$.*
- *$A'_{G'}$ finds all possible sets of round keys, if more than one set exists.*
- *Any expanded-key bits used in the round function of r consecutive rounds of G' can take on the same values as the expanded-key bits used in the round functions of G.*
- *If G contains initial and end-of-round whitening, any expanded-key bits used for the leftmost b bits of each whitening step in r consecutive rounds of G' can take on the same values as the whitening bits in G.*

Before beginning the proof, we have a few comments on the theorem and assumptions. We first note that for an attack on G' to be computationally feasible, it must involve $< 2^b$ (plaintext, ciphertext) pairs because otherwise an exhaustive search on G would

be possible, implying G is insecure against practical attacks. The first assumption is typical of existing block ciphers and is true of the elastic versions of block ciphers. The second assumption is true of block ciphers used in practice. The last two assumptions mean that the key schedule of G' is defined such that a subset of the expanded-key bits can have the same values as if they were generated by the key schedule of G. These assumptions are easily satisfied in practice by using the key schedule of G to generate a subset of the round key bits and a separate algorithm to generate the expanded-key bits required in G' for the additional $r' - r$ rounds and any whitening present in G' that is not present in G. Another option is if the key schedule of G' generates pseudorandom expanded-key bits such that it is possible the expanded-key bits for the round function and leftmost b bits of whitening in r consecutive rounds can take on the same values generated by the key schedule of G. In practice, given an expanded-key, it is feasible to check if the expanded-key adheres to a specific block cipher's key schedule. A subset of the expanded-key bits being tested can be inserted into the key schedule to generate additional key bits which can be checked against the bits in the value being tested.

The theorem holds by default for the case when $y = 0$, since G' is just G (with the possible addition of whitening which can be set to 0's when applying the attack if G does not contain whitening). We view G as having whitening steps in the proof to Theorem 1. This is not an issue for the following reason. If the attack on G' involves solving for the round key bits directly and allows the bits used in the whitening steps to be set to 0 for bit positions not swapped and to 0 or 1, as necessary, for bit positions swapped, then the whitening on the leftmost b bits is equivalent to XORing with 0, which is the same as having no whitening in G. If the attack on G' finds all possible keys or sets of round keys, the attack must find the key(s) or set(s) of round keys corresponding to round keys that are equivalent to XORing with 0. Setting a subset of bits in each whitening step in G' to 0's is equivalent to using a weaker version of G'. Any attack that works on G' will work on the weaker version. This is merely the case where the attacker knows certain bits of each whitening step are 0's.

We note that Theorem 1 only states that an attack on G' can be converted to an attack on G and not the reverse. This is because, in general, the claim that an attack on G can be converted into an attack on G' does not hold. Consider the case when G contains the initial and end-of-round whitening steps. When $y = 0$, G' is G with the initial and final key-dependent permutations added and the key schedule replaced (such as by a stream cipher). If the attack on G is due to the original key schedule, the attack does not necessarily hold if the key schedule is changed to generate pseudorandom bits when creating G'. For any attack not due to the key schedule, in order to claim that an attack on G implies an attack on G', it is necessary that the attack on G be such that the addition of the initial and final key-dependent permutations, the addition or expansion of the whitening steps and the addition of the swap steps do not result in the attack becoming inapplicable or computationally infeasible. In general, the conversion of an attack from G' to G works because there is a decrease in the complexity of the block cipher being attacked when going from G' to G; whereas, the reverse is not true because there is an increase in the complexity of the block cipher when converting G to G'.

To prove Theorem 1, we must show for any value of y, where $0 \leq y \leq b$, that if an attack exists on G' it can be converted into an attack on G using polynomial time

and memory. We define the steps for converting a round-key recovery attack on G' to an attack on G. We describe two ways of performing the conversion. The first method works for any value of y, where $0 \leq y \leq b$. The second method is is applicable for values of y satisfying $r(y-2) < b$, where r is the number of rounds in the original cipher. We include the second method because it requires fewer computations than the first method and thus is useful for small values of y. The methods treat whitening key bits as if they are pseudorandom in that the whitening key bits can take on any value. In G, if there is a relationship amongst the whitening key bits and/or between whitening key bits and key material used within the round function due to the key schedule of G, such keys will be a subset of all the possible sets of round keys found using the attack on G'. Then the set of round keys that satisfies the key schedule of G can be determined by checking which of the potential keys corresponds to the key schedule. If the number of potential sets of round keys found by the attack on G' is large enough such that it is computationally infeasible to determine which ones adhere to the key schedule of G, then the attack on G' is not computationally feasible. This is because the number of potential sets of round keys it finds for a set of (plaintext, ciphertext) pairs will also be large enough such that it is computationally infeasible for an attacker to determine which set to use to decrypt additional ciphertexts.

When we refer to converting the round keys of G' into round keys for G, we mean the following: In round j of G', let b_{jl} denote the l^{th} bit of the b bits output from the round function prior to the end-of-round whitening. Let kw_{jl} denote the end-of-round whitening key bit applied to b_{jl}. If b_{jl} is involved in the swap step at the end of round j, let y_{jh} denote the bit from the rightmost y bits with which b_{jl} is swapped and let kw_{jh} denote the whitening key bit applied to y_{jh}. Set the l^{th} whitening bit in round j of G to $kw_{jl} \oplus kw_{jh} \oplus y_{jh}$ when $j \geq 2$. When $j = 1$, the l^{th} whitening bit is set to $kw_{1l} \oplus kw_{1h} \oplus y_{1h} \oplus kw_{0h}$ in order to include the initial whitening on the rightmost y bits in the conversion. Set all other key bits used in G (both whitening and any internal to the round function) to be identical to the key bits used in G'. We refer to the initial whitening as round 0. The initial whitening for G' is converted to initial whitening for G by using the leftmost b expanded-key bits of the initial whitening as the initial whitening in G.

Proof of Theorem I: First Method. We describe here a method for converting the attack on G' to an attack on G. Without loss of generality, we use the first r rounds of G' as the r consecutive rounds for which the round keys are found. The conversion is presented in terms of solving for the round keys from the initial whitening to round r, but may also be performed by working from round r back to the initial whitening or by using any consecutive r rounds with whitening applied before the first round as long as the plaintext for G is the leftmost b bits of input to the r rounds and the corresponding ciphertext from G is the leftmost b bits of the output of the r rounds.

This attack runs in quadratic time in the number of rounds of G. The attack, $A'_{G'}$, on G' is used to solve for round keys 0 and 1 for G, then repeatedly solves for one round key of G at a time, using the output of one round of G as partial input to a reduced round version of G', running the attack on G' and converting the 1^{st} round key of G' to the round key for the next round of G. By the second condition in Theorem 1, if an attack on G' with r' rounds exists, then a reduced round attack on G' exists for any number of rounds $< r'$.

Let P be a set of plaintexts and C be a set of ciphertexts. We use the notation $\{(P, C)\}$ to indicate a set of (plaintext,ciphertext) pairs of the form (pi, ci) with $pi \in P$ and $ci \in C$. Given a set $\{(P^*, C^*)\} = \{(pi^*, ci^*)\}$ of n (plaintext, ciphertext) pairs for G, create a set $\{(P, C)\} = \{(pi^* \parallel 0, ci^* \parallel vi_r)\}$ of n (plaintext, ciphertext) pairs for an r-round version of G'. Note: we only require that the y bits appended to each pi^* when forming $\{(P, C)\}$ be a constant; we choose to use 0. The vi_r values appended to the ci^* values are arbitrary and do not need to be identical. The r subscript in vi_r denotes the number of rounds. Our method runs reduced round attacks on G' and the vi_r's can vary each time. Solve G' for round keys 0 and 1. By the pseudorandomness of the round keys, sets of round keys exist that correspond to $\{(P, C)\}$ and which are identical in at least the initial whitening and first round (the round keys across all n pairs may be identical in additional rounds, but we are only concerned with the initial whitening and first round at this point in the process). Denote these as rk'_0 and rk'_1. Use the leftmost b bits of rk'_0 as round key 0, rk_0, for G. Since the rightmost y bits are identical across all inputs to G', when rk'_1 is converted to a round key for G, the result will be the same across all n elements of $\{(P^*, C^*)\}$. Use the converted round key as round key 1, rk_1, for G. For each pi^*, apply the initial whitening and first round of G using the two converted round keys. Let pi_1 denote the output of the first round of G for $i = 1$ to n. Using a reduced round version of G' with $r - 1$ rounds and the initial whitening removed, set $\{(P, C)\} = \{(pi_1 \parallel 0, ci^* \parallel vi_{r-1})\}$ and solve for the first round key of G'. As before, convert the resulting round key for the first round of G' to a round key for G, but this time use the converted key as the second round key for G. Repeat the process for the remaining rounds of G, each time using the outputs of the last round of G for which the round key has been determined as the inputs to G' and reducing the number of rounds in G' by 1, to sequentially find the round keys for G.

This attack involves applying each round of G to n inputs for a total of rn rounds of G. $\frac{n(r+1)r}{2}$ rounds of G' are computed in the worst case if $A'_{G'}$ requires knowing the output of each round of the reduced round version of G' to find the first round key. r applications of $A'_{G'}$ are needed on the reduced round versions of G'. Let t_A denote the time to run $A'_{G'}$. Let ks_t be the time to check that an expanded-key found by $A'_{G'}$ adheres to the key schedule of G. The time to attack G is $O(nr^2 + rt_A + ks_t)$.

In summary, the attack on G can be written as:

Input $\{(P^*, C^*)\} = \{(pi^*, ci^*)$ for $i = 1$ to $n\}$.
Create $\{(P, C)\} = \{(pi^* \parallel 0, ci^* \parallel vi_r)$ for $i = 1$ to $n\}$ for a r-round version of G',
    where the $vi's$ are arbitrary.
Using $A'_{G'}$, solve a r-round version of G' for rk'_0 and rk'_1.
Convert rk'_0 to rk_0 and rk'_1 to rk_1.
Set $pi_1 =$ first round output of G using rk_0 and rk_1, for $i = 1$ to n.
For $j = 1$ to $r - 1$ {
        $\{(P, C)\} = \{(pi_j \parallel 0, ci^* \parallel vi_{r-j})$ for $i = 1$ to $n\}$.
        Solve a $r - j$ reduced round version of G' for the first round key, rk'_1.
        Convert rk'_1 to form rk_{j+1}.
        $pi_{j+1} =$ output of round $j + 1$ of G on pi_j using rk_{j+1}, for $i = 1$ to n.
}

Proof of Theorem I: Second Method. Our second method for proving Theorem 1 requires fewer computations than the first method, but provides round keys for a smaller set of (plaintext, ciphertext) pairs. The attack works as follows: Assume there exists a known (plaintext, ciphertext) pair attack on G' which produces the round keys either by finding the original key and then expanding it, or by finding the round keys directly. Using round keys for rounds 0 to r of G', convert the round keys into round keys for G one round at a time. For each round, extract the largest set of (plaintext, ciphertext) pairs used in the attack on G' that have the same converted round key. If there are n_j (plaintext, ciphertext) pairs involved at round j, there will be at least $\frac{n_j}{2^y}$ pairs remaining for which the round keys are consistent after round j. The end result is a set of round keys for G that are consistent with a set of $\frac{n}{2^{y(r-2)}}$ b-bit (plaintext, ciphertext) pairs for G. We then describe how to take a set of (plaintext, ciphertext) pairs for G, convert them into a set of (plaintext, ciphertext) pairs for G' in order to run the attack on G' to find the round keys for G.

Let $\{(P, C)\} = \{(pi \parallel xi, ci \parallel zi)\}$, for $i = 1$ to n, denote a set of n known $(b+y)$-bit (plaintext, ciphertext) pairs for G', where $|pi| = |ci| = b$ and $|xi| = |zi| = y$.

Let $A_{G'}$ be an attack on G' that finds the key(s) corresponding to $\{(P, C)\}$ in time less than an exhaustive search for the key. Let m denote the number of keys found. In practice, only one key should be found for any set of (plaintext, ciphertext) pairs. $m > 1$ only impacts the time to perform the attack and not the method itself. Without loss of generality, it is assumed that the keys are available in expanded form.

Let k be one of the m keys found by $A_{G'}$ and let ek be the expanded-key bits corresponding to k. Let $e\hat{k}_i$ be the expanded-key bits for G resulting from the conversion of ek when applied to the i^{th} element of $\{(P, C)\}$. Let R_{int} denote any bits of ek utilized within the round function. The values found for the bits of R_{int} will be the same for G' and G (the same in ek and every $e\hat{k}_i$). For each i, the bits of $e\hat{k}_i$ corresponding to the initial whitening in G (round 0) will be the leftmost b bits of the initial whitening bits from ek.

Let $\{(P, U)\} = \{(pi||xi, ui||vi)\}$ such that $ui||vi$ is the output of the r^{th} round of G' prior to the swap step, where $|ui| = b$ and $|vi| = y$.

When the round keys from ek are converted to those for $e\hat{k}_i$, at most y bits change in the leftmost b bits of each end-of-round whitening step. Thus, the resulting round keys for round q, $1 \leq q \leq r$ can be divided for each of the y impacted bits into those that have a 0 in the affected bit and those that have a 1 in the affected bit. For $q = 1$ to r, define S_{rnd_q} as the maximum-sized set of $e\hat{k}_i$ values from $S_{rnd_{q-1}}$ that have identical bits for round q, where $S_{rnd_0} = \{e\hat{k}_i$, for $i = 1$ to $n\}$. Let $\{(P, U)_{rnd_q}\}$ be the corresponding elements of $\{(P, U)\}$. When forming $\{(P, U)_{rnd_q}\}$, at least $(2^{-y}) * |\{(P, U)_{rnd_{q-1}}\}|$ of the elements from $\{(P, U)_{rnd_{q-1}}\}$ are included. There is no swap step after the r^{th} round so $|S_{rnd_r}| = |S_{rnd_{r-1}}|$. Across r rounds, the number of (plaintext, ciphertext) pairs are reduced at most $r - 1$ times.

To illustrate how the sets S_{rnd_q} and $\{(P, U)_{rnd_q}\}$ are created, consider the example shown in Figure 4 where $b = 4$, $y = 2$, and the leftmost 2 bits are swapped with the y bits in the swap step. The round number is q and $\{(P, U)_{rnd_{q-1}}\}$ contains three (plaintext, ciphertext) pairs. Suppose the outputs of the round function in the q^{th} of G' are $100101, 110011$ and 111111 and the whitening bits in the q^{th} round are 011010.

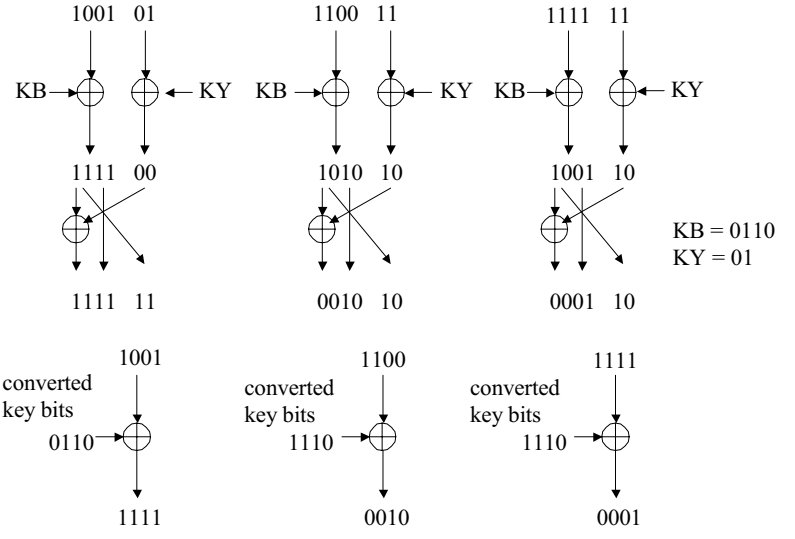


Fig. 4. Forming S_{rnd_q}

The whitening bits of the converted round keys corresponding to the three cases are $0110, 1110$ and 1110. Since 1110 occurs in the majority of the cases, set the q^{th} round key of G to 1110. S_{rnd_q} contains the elements of $S_{rnd_{q-1}}$ that produced 1110 as the q^{th} round key, and $\{(P,U)_{rnd_q}\}$ contains the second and third (plaintext, ciphertext) pairs from $\{(P,U)_{rnd_{q-1}}\}$.

Let rk be the contents of S_{rnd_r}. rk is the expanded key bits for G. Let $\{(P,C)_G\} = \{(pi,ci)|(pi \parallel yi, ui \parallel vi) \in \{(P,U)_{rnd_r}\}\}$. $|\{(P,C)_G\}| \geq n/2^{y(r-1)}$. $\{(P,C)_G\}$ is a set of (plaintext, ciphertext) pairs for which $G_{rk}(pi) = ci \ \forall \ (pi,ci) \in \{(P,C)_G\}$.

So far we have defined a method that produces a set of at least $\frac{n}{2^{y(r-1)}}$ (plaintext, ciphertext) pairs that are consistent with the round keys. This lower bound on the number of (plaintext, ciphertext) pairs can be slightly increased to $\frac{n}{2^{y(r-2)}}$ by using $(b+y)$-bit plaintexts that are the same in the rightmost y bits (which we did by setting these bits to 0). This will result in $|S_{rnd_1}| = n$. Since we also have $|S_{rnd_r}| = |S_{rnd_{r-1}}|$, the set of (plaintext, ciphertext) pairs is not reduced in the first and r^{th} rounds. Then the number of (plaintext, ciphertext) pairs produced for G that are consistent with the round keys for G is $\geq \frac{n}{2^{y(r-2)}}$. The number of possible plaintexts for G is 2^b; therefore, it is necessary for $y(r-2) < b$ to use this method.

To perform the attack on G when given a set of (plaintext, ciphertext) pairs for G, convert the pairs into a set of (plaintext, ciphertext) pairs for G' and find the round keys for G', and then for G as follows: Given a set $\{(P^*,C^*)\} = \{(pi^*,ci^*)\}$ for $i = 1$ to n known (plaintext, ciphertext) pairs for G, create the set $\{(P,C)\}$ of (plaintext, ciphertext) pairs to use in the attack on an r-round version of G' by setting $pi \parallel xi = pi^* \parallel 0$ and $ci \parallel zi = ci^* \parallel zi$, for $i = 1$ to n. For the set of (P,C) pairs created, $\{(P,U)\} = \{(pi^* \parallel 0, ci^* \parallel zi)\}$. Apply the attack on G' to solve for the round keys of G' then produce the sets $\{(P,U)_{rnd_r}\}$ and S_{rnd_r}. The round keys in S_{rnd_r} will be

consistent with the (plaintext, ciphertext) pairs in $\{(P,U)_{rnd_r}\}$. A set of round keys that adheres to the key schedule of G will be found by Claim 1 and the assumption that the attack on G' finds all possible sets of round keys.

Let t_r be the time to run r rounds of G' and t_A be the time to run $A_{G'}$. Recall that m is the number of keys (sets of round keys) found by $A'_{G'}$. In the case of obtaining at least one set $\{(P,U)_{rnd_r}\}$ of size $\geq \frac{n}{2^{y(r-2)}}$, the time required beyond t_A consists of nmt_r time to obtain the outputs of the first r rounds for each $\{(P,U)\}$, $O(nmr)$ time to perform the conversion of the round keys from G' to round keys for G and $O(nmr)$ time to form the S_{rnd_r} sets. Let ks_t be the time to check that an expanded-key adheres to the key schedule of G. Thus, the additional time required to attack G (beyond the time required to attack G') is $O(nm(r+t_r)+mks_t)$. The only unknown value is m. If m is large enough, to the extent that it approaches the average number of keys to test in a brute force attack on G', then this contradicts the assumption that an efficient attack exists on G' because the attacker is left with a large set of potential keys for decrypting additional ciphertexts.

4 Conclusion

We have proven that the elastic version of a block cipher is secure against any practical attack that attempts to recover key or expanded-key bits if the original cipher is secure against the attack. This eliminates the need to analyze an elastic version of a block cipher against these types of attacks if the original cipher is secure against such attacks (unless one is interested in improving the concrete work factors and probabilities of success). Our result follows from the network structure used in creating elastic block ciphers and the fact that the round function of the original fixed-length block cipher is used as a black box when forming its elastic version. We note that while reduction-based proofs of security are a cornerstone of cryptographic analysis, they are typical when complete components are used as sub-components in a larger design and used in a black box fashion. We are not aware of the use of such techniques in the case of concrete block cipher designs.

Acknowledgments

This work was partially supported by NSF Grants ITR CNS-04-26623 and CPA CCF-05-41093. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or the U.S Government.

References

1. Bellare, M., Rogaway, P.: On the Construction of Variable Length-Input Ciphers. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 231–244. Springer, Heidelberg (1999)
2. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994)

3. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, New York (1993)
4. Cook, D.: Elastic Block Ciphers, Ph.D. Thesis, Columbia University (2006)
5. Cook, D., Yung, M., Keromytis, A.: Elastic Block Ciphers: The Basic Design. In: Proceedings of ASIACCS, pp. 350–355. ACM, New York (2007)
6. Knudsen, L.: Block Ciphers - Analysis, Design and Applications, Ph.D. Thesis, Aarhus University (1994), http://www2.mat.dtu.dk/people/Lars.R.Knudsen
7. Knudsen, L.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) Fast Software Encryption. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
8. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. Siam Journal of Computing 17(2), 373–386 (1988)
9. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
10. NIST, FIPS 197 Advanced Encryption Standard (AES) (2001)
11. Patel, S., Ramzan, Z., Sundaram, G.: Efficient Constructions of Variable-Input-Length Block Ciphers. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 326–340. Springer, Heidelberg (2004)
12. Vaudenay, S.: A Classical Introduction to Cryptography. Springer, Berlin (2006)
13. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

Impossible-Differential Attacks on Large-Block Rijndael

Jorge Nakahara Jr. and Ivan Carlos Pavão

UNISANTOS, Brazil
jorge_nakahara@yahoo.com.br, icpeapla@yahoo.com.br

Abstract. This paper[1] reports impossible-differential (ID) attacks on reduced-round versions of the Rijndael cipher with text blocks larger than 128 bits. These attacks follow the framework of the attacks by Biham-Keller and Cheon *et al.* on the AES, and reach up to seven rounds of large-block Rijndael variants. Even though these ciphers are not standardized as FIPS, like the AES, it is important to evaluate the security of the other Rijndael versions since they provide larger internal states when used as primitives for the construction of stream ciphers and hash functions. The main contributions of this paper are longer ID distinguishers found for large-block Rijndael versions, compared to the ones used for the AES.

Keywords: Impossible differential cryptanalysis, information security.

1 Introduction

Rijndael is a Substitution Permutation Network (SPN) block cipher designed by Joan Daemen and Vincent Rijmen for the AES Development Process [1,9], initiated by the National Institute of Standards and Technology (NIST) in the USA in 1997. In Rijndael, both the text block and the key sizes can range from 128 up to 256 bits in steps of 32 bits [10, p.42]. Rijndael is an iterated cipher. The number of rounds, Nr, depends on the text block and the key sizes, Nk, in steps of 32 bits. There are 25 instances of Rijndael [10,18], for all combinations of key and text block sizes. The 128-bit block version of Rijndael, with a key of 128, 192 or 256 bits, is officially known as the AES [12]. We denote by Rijndael-b, with $b \in \{160, 192, 224, 256\}$, the large-block Rijndael versions with the suffix indicating the text block size in bits. In [1], the blocks for text, key, subkeys and intermediate data are represented compactly by a $4 \times Nb$ state matrix of bytes, where Nb is the number of 32-bit words in a block. For example, the state matrix for a $4t$-byte text block, $A = (a_0, a_1, a_2, a_3, a_4, \ldots, a_{4t-1})$, is denoted

$$\text{State} = \begin{pmatrix} a_0 & a_4 & \ldots & a_{4t-4} \\ a_1 & a_5 & \ldots & a_{4t-3} \\ a_2 & a_6 & \ldots & a_{4t-2} \\ a_3 & a_7 & \ldots & a_{4t-1} \end{pmatrix}, \tag{1}$$

with bytes inserted columnwise. In AES, $t = 4$.

[1] Research funded by FAPESP under contract number 2005/02102-9.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 104–117, 2007.

Each (full) round in Rijndael contains four layers: SubBytes (denoted SB), ShiftRows (SR), MixColumns (MC) and AddRoundKey (AK_i), where $0 \leq i \leq Nr$ is the round number. We briefly explain each layer. More details can be found in [9,10]. SB is the nonlinear layer and provides the confusion [26] property in Rijndael. SB consists of the parallel application of a fixed 8×8-bit S-box to each byte of the state. Both SR and MC provide diffusion [26] in Rijndael. SR consists of left-rotating the rows of the state by fixed offsets. MC consists of a linear transformation over the columns of the state, using a 4×4 MDS matrix[1]. AK_i consists of the exclusive-or combination of the i-th round subkey with the intermediate cipher state. AK_i is the only key-dependent transformation in a round. One (full) round of Rijndael, operating on a text block X, can be denoted $F(X) = AK_i \circ \text{MC} \circ \text{SR} \circ \text{SB}\ (X) = AK_i(MC(SR(SB(X))))$. There is a pre-whitening layer, consisting of AK_0 only, before the first round. Moreover, the last round does not contain MC.

This paper applies the impossible differential (ID) technique [3,15] to up to seven rounds of Rijndael with text blocks larger than 128 bits, complementing the analysis in [6,8]. Even though these Rijndael variants have not been standardized in a FIPS, like the AES, some of them have attracted attention. For example, Rijndael-256 had its software performance [23, p.55] evaluated by the NESSIE Project [22], but there was no security analyses. We look at filling this gap, since block ciphers are pervasive cryptographic primitives, suitable as a building block in stream ciphers, hash functions and MAC constructions [20]. Weaknesses in the former may have negative consequences for the latter. Examples of applications of block ciphers include the OFB, CFB and CTR modes of operation (with security related to the size of the internal cipher state), and the Davies-Meyer, Miyaguchi-Preneel and Matyas-Meyer-Oseas constructions [20, p.340], where the hash size depends on the block size of the underlying cipher. The larger block sizes of Rijndael are in line with the new SHA-2 hash functions (SHA-224, SHA-256, SHA-384, SHA-512) [11].

Typical ID distinguishers, following the miss-in-the-middle technique, are constructed from two truncated differentials, one of them propagating in the encryption direction and the other in the decryption direction. Thus, ID distinguishers exploit both ends of a cipher at the same time, like boomerang [7] and differential-linear [17] distinguishers. Moreover, complete diffusion takes more rounds for large-block Rijndael variants. Consequently, ID distinguishers can be longer for increasing block sizes, and therefore, allowing attacks on a larger number of rounds. All of these facts motivate our security evaluation of large-block Rijndael.

This paper is organized as follows. Sect. 2 gives a brief overview of the impossible differential technique. Sect. 3 describes ID distinguishers and attacks on reduced-round versions of Rijndael-160. Sect. 4 describes ID distinguishers and attacks on versions of Rijndael-192. Sect. 5 describes ID distinguishers and attacks on versions of Rijndael-224. Sect. 6 describes ID distinguishers and attacks on versions of Rijndael-256. Sect. 7 concludes the paper.

[1] Maximum Distance Separable [1].

2 Impossible Differential Attacks

Distinguishers are fundamental tools in cryptanalysis, and can be used either to distinguish a cipher from a random permutation, or in key-recovery attacks. Informally, distinguishers help detect nonrandom behaviors of a given cipher, such as nonuniform distribution of text difference patterns (in DC, ID [15] and boomerang [7] analyses), biased linear relations (in LC [19]) or algebraic properties (algebraic and interpolation [14] analyses). Distinguishers can be key-independent, if they hold for any key value, or key-dependent, if they hold only for particular key values, called weak keys. Unlike the differential and linear techniques, which look for events such as text patterns or statistical correlations of high probability (or high bias), the ID method looks for events that never happen. It is an open problem how to use ID distinguishers that hold with a negligibly small (but nonzero) probability, that is, probabilistic ID distinguishers.

The impossible differential (ID) technique is a chosen-plaintext (CP) attack formerly proposed in [15] against the DEAL block cipher, and further applied to Skipjack [3], IDEA [4], Khufu [4], the AES [6] and many other ciphers. ID distinguishers currently reported in the literature use the miss-in-the-middle technique [3]. This technique requires two differentials (∇ and Δ) both holding with certainty (probability one). In ∇, the difference patterns propagate in the encryption direction. In Δ, the difference patterns propagate in the decryption direction. Both differentials are constructed such that the output difference pattern of ∇ is incompatible with the input difference pattern of Δ, in the sense that the output difference of ∇ cannot cause the output difference of Δ (and vice-versa). This contradiction explains the term miss-in-the-middle, and it is denoted $\nabla \not\rightarrow \Delta$. Symmetrically, $\Delta \not\rightarrow \nabla$ (the latter works in a chosen-ciphertext (CC) setting).

In byte-oriented ciphers such as Rijndael (AES), it is typical to use truncated differentials [16] to construct Δ and ∇, because truncated difference patterns hold with certainty. Moreover, they are also independent of the particular S-boxes used in the cipher. In truncated differentials, one only distinguishes between zero and nonzero differences, namely, the exact value of the nonzero difference is irrelevant. For bytewise difference patterns, as in Rijndael, a nonzero byte difference will be denoted 'δ'. In contrast, a zero byte difference will be denoted simply '0'. Notice that although δ is used throughout the distinguisher, it does not mean that all these bytes contain the same difference value. It only means that the difference value in nonzero. The difference operator used for Rijndael is exclusive-or.

Differential and linear distinguishers (among others) recognize the correct key by comparing difference patterns or linear relations that most closely satisfy the distinguishers. ID distinguishers operate the other way around. The keys that actually satisfy the ID distinguisher are wrong values, and the (single) key value not suggested is the correct one.

ID distinguishers in this paper are independent of the key schedule algorithm, and of weak-key or weak-subkey assumptions. Even though modern variants

of the impossible differential attacks on the AES [5,13,27] operate under related-key assumptions, the same technique cannot be applied to the other Rijndael variants, because no key schedule has been officially defined for the latter [25]. Furthermore, in all attacks we assume the user key size equals the block size.

3 ID Distinguisher for Rijndael-160

An example distinguisher for 4-round Rijndael-160 is described in (2) in which the ∇ truncated differential covers AK_0 until MC of the third round. The Δ truncated differential covers AK_4 up until MC of the third round. These two differentials are incompatible because of the pattern of three nonzero byte differences in the leftmost column of the state before the MC layer of the third round, and the pattern of four zero byte differences in the same column after the MC layer. Since the branch number of the MC matrix is five [9,10], these difference patterns are contradictory. Difference propagation is denoted by the symbol $\rightarrow$, and means that the difference pattern on the left-hand side causes the difference pattern on the right-hand side.

$$
\begin{pmatrix} \delta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\overset{MC \circ SR \circ SB \circ AK_0}{\rightarrow}
\begin{pmatrix} \delta & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 \end{pmatrix}
\overset{SR \circ SB \circ AK_1}{\rightarrow}
$$

$$
\begin{pmatrix} \delta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta \\ 0 & 0 & 0 & \delta & 0 \\ 0 & 0 & \delta & 0 & 0 \end{pmatrix}
\overset{SB \circ AK_2 \circ MC}{\rightarrow}
\begin{pmatrix} \delta & 0 & \delta & \delta & \delta \\ \delta & 0 & \delta & \delta & \delta \\ \delta & 0 & \delta & \delta & \delta \\ \delta & 0 & \delta & \delta & \delta \end{pmatrix}
\overset{SR}{\rightarrow}
\begin{pmatrix} \delta & 0 & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & 0 & \delta \end{pmatrix}
\overset{MC}{\not\rightarrow}
\begin{pmatrix} 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \end{pmatrix}
$$

$$
\overset{AK_3^{-1} \circ SB^{-1}}{\leftarrow}
\begin{pmatrix} 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta \end{pmatrix}
\overset{SR^{-1} \circ AK_4^{-1}}{\leftarrow}
\begin{pmatrix} 0 & \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & 0 & \delta \\ \delta & \delta & 0 & \delta & \delta \end{pmatrix}
\tag{2}
$$

Distinguisher (2) belongs to a set of related distinguishers, all of which share the same input difference pattern, the same number of rounds, and the same number of zero output byte differences, but the precise output difference pattern changes. For example, (2) contains zero byte differences in the ciphertext positions (0,11,14,17) of the state matrix (1), but other ciphertext difference patterns also cause contradiction. These patterns contain zero byte differences in ciphertext positions (1,4,15,18), (2,5,8,19), (3,6,9,12) or (7,10,13,16). Thus, all of these ciphertext byte positions are forbidden because all of them lead to a contradiction.

There are many different ID distinguishers. But, some aspects make a distinguisher significantly more useful for a effective attack. Primarily, the number of nonzero input byte differences to ∇ is related to the total amount of plaintext pairs that can be generated and processed (with m texts one can make $m(m-1)/2$ text pairs). This number might not be too large, otherwise, the attack may require too many encryptions. On the other hand, it should also not be too small, otherwise, no right pairs (text pairs that follow the ID distinguisher

patterns) will survive filtration by the ciphertext difference pattern. Further, the number of zero ciphertext byte differences indicates the expected number of pairs that may satisfy the ciphertext difference. The more zero ciphertext byte differences, the larger the number of filtered pairs. Ideally, this number shall be small.

3.1 Attack on 5-Round Rijndael-160

Distinguisher (2) can be used to recover AK_0 piecewise in an attack on 5-round Rijndael-160. The distinguisher is placed in the last four rounds. The attack works as follows:

(i) create a pool of 2^{32} plaintexts $P_i = (p_0^i, p_1^i, \ldots, p_{19}^i)$, $0 \leq i < 2^{32}$, such that $(p_0^i, p_5^i, p_{10}^i, p_{15}^i)$ range over all 32-bit values, while the remaining bytes assume arbitrary constant values. Encrypt this pool across 5-round Rijndael-160, and obtain a corresponding ciphertext pool $C_i = (c_0^i, \ldots, c_{19}^i)$;
(ii) for each pair of ciphertexts (C_i, C_j), $i \neq j$, such that the byte differences at positions (0,11,14,17) of the state (1) are zero, guess 32 bits of AK_0 in positions (0,5,10,15) and decrypt the first round of (P_i, P_j) up to the leftmost column of the first MC layer. If only one byte difference is nonzero in this column, then the guessed 32-bit subkey is wrong;
(iii) output the (only) 32-bit subkey value that was not eliminated in (ii).

Step (i) creates a pool of 2^{32} ciphertext C_i, and a total of $2^{32}(2^{32}-1)/2 \approx 2^{63}$ pairs (C_i, C_j) in (ii). The forbidden patterns in $C_i \oplus C_j$, include the four zero byte differences in positions (0,11,14,17), (1,4,15,18), (2,5,8,19), (3,6,9,12) and (7,10,13,16) of the state matrix (1). Thus, the expected number of ciphertext pairs that satisfy any of these forbidden patterns is $\frac{2^{63}}{5\cdot(2^8)^4} \approx 2^{29}$. So, we expect to test about 2^{29} pairs in step (ii). Each 32-bit subkey candidate that satisfies step (ii) lead to a column of the state with three zero byte differences. Thus, one expects that $2^{32} \cdot (2^{-8})^3 = 2^8$ wrong subkey values are suggested per pair.

After one plaintext pool is processed, the expected number of wrong subkeys remaining is $2^{32}(1-\frac{2^8}{2^{32}})^{2^{29}} = 2^{32}(1-2^{-24})^{2^{29}} \approx 2^{32}(e^{-1})^{2^5} = \frac{2^{32}}{e^{32}} < 1$, so only the correct subkey value remains.

The complexity of step (i) is 2^{32} 5-round encryptions. In step (ii), each pair is partially decrypted for 2^{32} subkey candidates, which is equivalent to $2^{32} \cdot 2^{29} = 2^{61}$ one-round decryptions. In order to recover the full AK_0, the same procedure must be repeated five times for each 32-bit piece of AK_0. Namely, (2) has nonzero (plaintext) byte differences in position 0. To recover the remaining 128 user key bits, we simply repeat the attack, but using distinguishers for which the nonzero (plaintext) byte is in position 4, 8, 12 or 16. Thus, the total effort is $5\cdot 2^{61}/5 = 2^{61}$ 5-round computations. The data complexity is $5 \cdot 2^{32} \approx 2^{34.3}$ CP. The memory required include storage of 2^{32} ciphertexts, and about 2^{32} bits for keeping track of the wrong subkey candidates.

Since this attack recovers AK_0 (which is supposed to be the user key) there are no savings in attack complexity by exploiting potential redundancies in the key schedule of the cipher. This is the motivation for recoverying AK_0 and not AK_5.

An interesting question is: why does the correct subkey is not filtered by the ID distinguisher? In the given key-recovery attack, the wrong subkeys do not effectively decrypt the first round. Only the correct subkey can do it. Therefore, in the latter, what remains (after decryption) is the ID distinguisher itself, that can never be satisfied (in the case of a single active difference byte in a column of the state); if there is more than one active difference byte in a column, the distinguisher is definitely not satisfied at all. In the former, the wrong subkeys not only do not decrypt the first round, but also effectively add a round on top of the first round thus, further randomizing the input to the ID distinguisher.

3.2 Attack on 6-Round Rijndael-160

Another attack using (2) can recover subkey bits from both AK_0 and AK_6, on 6-round Rijndael-160, similar to [8]. This attack works as follows:

(a) create a pool of 2^{32} plaintexts $P_i = (p_0^i, \ldots, p_{19}^i)$ such that $(p_0^i, p_5^i, p_{10}^i, p_{15}^i)$ assume all possible 32-bit values, and the remaining bytes assume arbitrary constant values;

(b) consider $2^{73.5}$ pools, which mean $2^{105.5}$ chosen plaintexts (CP) and $2^{136.5}$ plaintext pairs. Find ciphertext pairs that contain zero byte difference in the bottommost two rows of the state matrix (a 2^{-80} condition). The expected number of pairs that satisfy this restriction is $2^{136.5-80} = 2^{56.5}$; there is no need to guess the key where the byte difference is zero because in these positions the text values are the same, and whatever the key value, the decrypted value will be equal (though unknown);

(c) guess 80 bits of $AK_6 = (k_{6,0}, k_{6,1}, \ldots, k_{6,19})$ corresponding to the topmost two rows of the state matrix, i.e. $(k_{6,0}, k_{6,1}, k_{6,4}, k_{6,5}, k_{6,8}, k_{6,9}, k_{6,12}, k_{6,13}, k_{6,16}, k_{6,17})$;

(d) for each ciphertext pair (C, C') that satisfies step (b), decrypt the last round and compute $\mathrm{MC}^{-1}(C \oplus C')$ and check if there are zero byte differences in one of the five forbidden positions (0,11,14,17), (1,4,15,18), (2,5,8,19), (3,6,9,12), (7,10,13,16) of the state matrix (1). The joint probability of these difference patterns is $5{\cdot}2^{-32} \approx 2^{-29.5}$, and the expected number of remaining pairs is $2^{56.5} \cdot 2^{-29.5} = 2^{27}$;

(e) for a plaintext pair (P, P') corresponding to a ciphertext pair from step (d), guess 32 subkey bits $(k_{0,0}, k_{0,5}, k_{0,10}, k_{0,15})$ of $AK_0 = (k_{0,0}, k_{0,1}, \ldots, k_{0,19})$ and decrypt the first round until after the MC layer. Keep those pairs for which there is only one nonzero byte difference in the leftmost column after MC. The probability of this event is $4 \cdot (2^8 - 1)/2^{32} \approx 2^{-22}$, for three zero byte differences in a single column of MC;

(f) every subkey that leads to such difference is wrong. After analyzing $2^{73.5}$ pools, there remains $2^{32}(1-2^{-22})^{2^{27}} \approx 2^{32} \cdot e^{-32} < 1$ wrong values for AK_0;

(g) steps (c) and (d) require $2^{56.5} \cdot 2^{80} = 2^{136.5}$ 1-round computations, and step (e) requires $2^{80} \cdot 2^{32}(1 + (1 - 2^{-22}) + (1 - 2^{-22})^2 + \ldots (1 - 2^{-22})^{2^{27}}) \approx 2^{134}$ 1-round computations. The total time complexity is $(2^{136.5} + 2^{134})/6 \approx 2^{134}$ 6-round computations, $2^{105.5}$ CP and 2^{80} bits or $2^{69.7}$ blocks of memory to recover 80 bits of AK_6 and 32 bits of AK_0. To recover the remaining 80 bits of AK_6, just repeat the same attack, but look for the subkey bits on the lower two rows of the state matrix. Thus, only the time complexity doubles.

4 ID Distinguisher for Rijndael-192

A 4-round ID distinguiser for Rijndael-192 is depicted in (3), covering AK_0 until AK_4.

$$\begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \overset{MC \circ SR \circ SB \circ AK_0}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \overset{SR \circ SB \circ AK_1}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta \\ 0 & 0 & 0 & 0 & \delta & 0 \\ 0 & 0 & 0 & \delta & 0 & 0 \end{pmatrix}$$

$$\overset{SB \circ AK_2 \circ MC}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & \delta & \delta & \delta \\ \delta & 0 & 0 & \delta & \delta & \delta \\ \delta & 0 & 0 & \delta & \delta & \delta \\ \delta & 0 & 0 & \delta & \delta & \delta \end{pmatrix} \overset{SR}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & \delta & \delta & \delta \\ 0 & 0 & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & \delta & 0 & 0 \end{pmatrix} \overset{MC}{\not\rightarrow} \begin{pmatrix} 0 & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta \end{pmatrix} \overset{AK_3^{-1} \circ SB^{-1}}{\leftarrow}$$

$$\begin{pmatrix} 0 & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta \end{pmatrix} \overset{SR^{-1} \circ AK_4^{-1}}{\leftarrow} \begin{pmatrix} 0 & \delta & \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & \delta & 0 & \delta \\ \delta & \delta & \delta & 0 & \delta & \delta \end{pmatrix} \tag{3}$$

A similar contradiction between difference patterns in (2) also happens in (3), before and after the MC layer of the third round. We describe in Appendix A an example of a 6-round ID distinguisher for Rijndael-192 that is longer than (3).

4.1 Attack on 5-Round Rijndael-192

An attack on 5-round Rijndael-192 using (3) is similar to the one on Rijndael-160 using (2). We use plaintext pools with 2^{32} plaintexts $P_i = (p_0^i, p_1^i, \ldots, p_{23}^i)$, such that $(p_0^i, p_5^i, p_{10}^i, p_{15}^i)$ range over all 32-bit values, while the remaining bytes assume arbitrary constant values. But now there are six ciphertext difference patterns (with zero byte differences) that are incompatible with the difference pattern in (3). They are (0, 15, 18, 21), (1, 4, 19, 22), (2, 5, 8, 23), (3, 6, 9, 12), (7, 10, 13, 16) and (11, 14, 17, 20).

Moreover, we have to recover 192 round subkey bits, so the attack has to be repeated six times, each of which recovers 32 bits of AK_0 at a time.

Analogous to the forbidden (ciphertext) positions for Rijndael-160, there are forbidden (plaintext) nonzero byte positions for Rijndael-192: 0, 4, 8, 12, 16, or 20. Apart from these details, the attack complexities (data/time/memory) for 5-round Rijndael-192 are computed similarly to Rijndael-160: $6 \cdot 2^{61}/5 \approx 2^{61}$ 5-round computations; $6 \cdot 2^{32} \approx 2^{34.6}$ CP; memory required is about 2^{32} text blocks.

4.2 Attack on 6-Round Rijndael-192

Another attack using (3) can recover subkey bits from both AK_0 and AK_6, on 6-round Rijndael-192, following [8]. This attack works as follows:

(a) create a pool of 2^{32} plaintexts $P_i = (p_0^i, p_1^i, \ldots, p_{23}^i)$ such that $(p_0^i, p_5^i, p_{10}^i, p_{15}^i)$ assume all possible 32-bit values, and the remaining bytes assume arbitrary constant values;

(b) consider $2^{89.5}$ pools, which mean $2^{121.5}$ chosen plaintexts (CP) and $2^{152.5}$ plaintext pairs. Find ciphertext pairs that contain zero byte difference in the bottommost two rows of the state matrix, which is a $(2^{-8})^{12} = 2^{-96}$ condition. The expected number of pairs that satisfy this restriction is $2^{152.5-96} = 2^{56.5}$;

(c) guess 96 bits of $AK_6 = (k_{6,0}, k_{6,1}, \ldots, k_{6,23})$ corresponding to the topmost two rows of the state matrix, i.e. $(k_{6,0}, k_{6,1}, k_{6,4}, k_{6,5}, k_{6,8}, k_{6,9}, k_{6,12}, k_{6,13}, k_{6,16}, k_{6,17}, k_{6,20}, k_{6,21})$;

(d) for each ciphertext pair (C, C') that satisfies step (b), decrypt the last round and compute $MC^{-1}(C \oplus C')$ and check if there are zero byte differences in one of the six forbidden positions (0,15,18,21), (1,4,19,22), (2,5,8,23), (3,6,9,12), (7,10,13,16) and (11,14,17,20). The joint probability of these difference patterns is $6 \cdot 2^{-32} \approx 2^{-29.4}$, and the expected number of remaining pairs is $2^{56.5} \cdot 2^{-29.4} = 2^{27.1}$;

(e) for a plaintext pair (P, P') corresponding to a ciphertext pair from step (d), guess 32 subkey bits $(k_{0,0}, k_{0,5}, k_{0,10}, k_{0,15})$ of $AK_0 = (k_{0,0}, k_{0,1}, \ldots, k_{0,23})$ and decrypt the first round until after the MC layer. Keep those pairs for which there is only one nonzero byte difference in the leftmost column after MC. The probability of this event is $4 \cdot (2^8 - 1)/2^{32} \approx 2^{-22}$ due to three zero byte differences in a single column of MC;

(f) every subkey that leads to such difference is wrong. There remains about $2^{32}(1 - 2^{-22})^{2^{27.1}} \approx 2^{32}/e^{2^{5.1}} < 1$ wrong values for AK_0;

(g) steps (c) and (d) require $2^{56.5} \cdot 2^{96} = 2^{152.5}$ 1-round computations, and step (e) requires $2^{96} \cdot 2^{32}(1 + (1 - 2^{-22}) + (1 - 2^{-22})^2 + \ldots (1 - 2^{-22})^{2^{27.1}}) \approx 2^{150}$ 1-round computations. The total time complexity is $(2^{152.5} + 2^{150})/6 \approx 2^{150}$ 6-round computations, $2^{121.5}$ CP and 2^{96} bits or $2^{85.4}$ blocks of memory to recover 96 bits of AK_6 and 32 bits of AK_0. To recover the remaining bits of AK_6, just repeat the same attack, but look for the subkey bits on the lower two rows of the state matrix. The time complexity doubles.

5 ID Distinguisher for Rijndael-224

An example of ID distinguisher for Rijndael-224 is (4), covering five rounds, from AK_0 until AK_5. The contradiction happens before and after the MC layer of the fourth round. The leftmost column before the third MC contains only one δ, while the remaining columns contain at least two δs. This mean that after

MC the state will contain only one column of δ's while the remaining columns will contain unpredictable differences (denoted by $*$), meaning that the byte differences could be either zero or nonzero. Before the fourth MC, the leftmost column contains at least one δ, that is, at least one nonzero value, while after MC, there are four zero byte differences. It contradicts the branch number of the MC matrix, which is five.

$$\begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \overset{MC \circ SR \circ SB \circ AK_0}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \overset{SR \circ SB \circ AK_1}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \delta \\ 0 & 0 & 0 & 0 & 0 & \delta & 0 \\ 0 & 0 & 0 & \delta & 0 & 0 & 0 \end{pmatrix}$$

$$\overset{SB \circ AK_2 \circ MC}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & \delta & 0 & \delta & \delta \\ \delta & 0 & 0 & \delta & 0 & \delta & \delta \\ \delta & 0 & 0 & \delta & 0 & \delta & \delta \\ \delta & 0 & 0 & \delta & 0 & \delta & \delta \end{pmatrix} \overset{SR}{\rightarrow} \begin{pmatrix} \delta & 0 & 0 & \delta & 0 & \delta & \delta \\ 0 & 0 & \delta & 0 & \delta & \delta & \delta \\ 0 & \delta & 0 & \delta & \delta & \delta & 0 \\ 0 & \delta & \delta & \delta & 0 & 0 & \delta \end{pmatrix} \overset{MC}{\rightarrow} \begin{pmatrix} \delta & * & * & * & * & * & * \\ \delta & * & * & * & * & * & * \\ \delta & * & * & * & * & * & * \\ \delta & * & * & * & * & * & * \end{pmatrix} \overset{SR \circ SB \circ AK_3}{\rightarrow}$$

$$\begin{pmatrix} \delta & * & * & * & * & * & * \\ * & * & * & * & * & * & \delta \\ * & * & * & * & * & \delta & * \\ * & * & * & \delta & * & * & * \end{pmatrix} \overset{MC}{\not\rightarrow} \begin{pmatrix} 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta & \delta \end{pmatrix} \overset{AK_4^{-1} \circ SB^{-1} \circ SR^{-1} \circ AK_5^{-1}}{\leftarrow} \begin{pmatrix} 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta & \delta & \delta & 0 \\ \delta & \delta & \delta & \delta & \delta & 0 & \delta \\ \delta & \delta & \delta & 0 & \delta & \delta & \delta \end{pmatrix} \quad (4)$$

We describe in Appendix A an example of a 6-round ID distinguisher for Rijndael-224 that is longer than (4), but ineffective for a key-recovery attack.

5.1 Attack on 6-Round Rijndael-224

An attack on 6-round Rijndael-224 using (4) is similar to the one on Rijndael-160 using (2). A plaintext pool contains 2^{32} plaintexts $P_i = (p_0^i, p_1^i, \ldots, p_{27}^i)$, such that $(p_0^i, p_5^i, p_{10}^i, p_{19}^i)$ range over all 32-bit values, while the remaining bytes assume arbitrary constant values. But, in this case there are only four ciphertext differences pattern that are incompatible with the plaintext difference pattern in (4), because of the four δs after the fourth MC layer in (4). These patterns are (0,15,22,25), (6,9,12,27), (7,14,17,20) and (11,18,21,24) of the state.

Moreover, we have to recover 224 user key bits, so the attack has to be repeated seven times, each of which recovers 32 user key bits of AK_0. The attack complexities (data/time/memory) for 6-round Rijndael-224 are: $7 \cdot 2^{61}/6 \approx 2^{61}$ 6-round computations; $7 \cdot 2^{32} \approx 2^{34.8}$ CP; memory required is 2^{32} text blocks.

5.2 Attack on 7-Round Rijndael-224

An attack on 7-round Rijndael-224, using (4), can be applied in a similar way as the attack on 6-round Rijndael-192.

The time complexity is 2^{167} 7-round computations, 2^{138} CP and 2^{112} bits or about 2^{104} blocks of memory, to recover the full AK_7 and 32 bits of AK_0.

6 ID Distinguisher for Rijndael-256

An example of ID distinguisher for Rijndael-256 is (5), covering five rounds, from AK_0 until AK_5. A similar contradiction between difference patterns as in (4) also happens in (5), before and after the MC layer of the fourth round. Notice

that before the fourth MC, the second column from the left contains $(*, \delta, *, *)$, where '$*$' can be either a zero or a nonzero difference. After MC, the same column contains only zero byte differences. Thus, there is at least one and at most four nonzero byte differences. Whatever the value of the '$*$'s, there is a contradiction due to the branch number of the MC matrix, which is five.

$$\begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{MC \circ SR \circ SB \circ AK_0} \begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{SR \circ SB \circ AK_1} \begin{pmatrix} \delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta \\ 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 \end{pmatrix}$$

$$\xrightarrow{SB \circ AK_2 \circ MC} \begin{pmatrix} \delta & 0 & 0 & 0 & \delta & \delta & 0 & \delta \\ \delta & 0 & 0 & 0 & \delta & \delta & 0 & \delta \\ \delta & 0 & 0 & 0 & \delta & \delta & 0 & \delta \\ \delta & 0 & 0 & 0 & \delta & \delta & 0 & \delta \end{pmatrix} \xrightarrow{SR} \begin{pmatrix} \delta & 0 & 0 & 0 & \delta & \delta & 0 & \delta \\ 0 & 0 & 0 & \delta & \delta & 0 & \delta & \delta \\ 0 & \delta & \delta & 0 & \delta & \delta & 0 & 0 \\ \delta & \delta & 0 & \delta & \delta & 0 & 0 & 0 \end{pmatrix} \xrightarrow{MC} \begin{pmatrix} * & * & \delta & * & * & * & \delta & * \\ * & * & \delta & * & * & * & \delta & * \\ * & * & \delta & * & * & * & \delta & * \\ * & * & \delta & * & * & * & \delta & * \end{pmatrix} \xrightarrow{SR \circ SB \circ AK_3}$$

$$\begin{pmatrix} * & * & \delta & * & * & * & \delta & * \\ * & \delta & * & * & * & \delta & * & * \\ * & * & * & \delta & * & * & * & \delta \\ * & * & \delta & * & * & * & \delta & * \end{pmatrix} \overset{MC}{\not\rightarrow} \begin{pmatrix} \delta & 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ \delta & 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ \delta & 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ \delta & 0 & \delta & \delta & \delta & \delta & \delta & \delta \end{pmatrix} \xleftarrow{AK_4^{-1} \circ SB^{-1} \circ SR^{-1} \circ AK_5^{-1}} \begin{pmatrix} \delta & 0 & \delta & \delta & \delta & \delta & \delta & \delta \\ 0 & \delta & \delta & \delta & \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta & \delta & \delta & 0 & \delta \\ \delta & \delta & \delta & \delta & \delta & 0 & \delta & \delta \end{pmatrix} \quad (5)$$

6.1 Attack on 6-Round Rijndael-256

An attack on 6-round Rijndael-256 using (5) is similar to the one on Rijndael-224 using (4). A plaintext pool contains 2^{32} plaintexts $P_i = (p_0^i, p_1^i, \ldots, p_{31}^i)$, such that $(p_0^i, p_5^i, p_{14}^i, p_{19}^i)$ range over all 32-bit values, while the remaining bytes assume arbitrary constant values. But, in this case, there are six ciphertext difference patterns that are incompatible with the plaintext difference pattern in (5). They are (1,4,23,26), (5,8,27,30), (2,9,12,31), (7,10,17,20), (11,14,21,24) and (15,18,25,28) of the state (1).

Moreover, we have to recover 256 user key bits, so the attack has to be repeated eight times, each of which recovers 32 bits of AK_0. The attack complexities (data/time/memory) are: $8 \cdot 2^{61}/6 \approx 2^{61.4}$ 6-round computations; $8 \cdot 2^{32} = 2^{35}$ CP; memory required is 2^{32} text blocks.

6.2 Attack on 7-Round Rijndael-256

An attack on 7-round Rijndael-256 using (5) can be applied in a similar setting as the attack on 7-round Rijndal-224.

The time complexity is 2^{182} 7-round computations, 2^{153} CP and 2^{128} bits or 2^{117} block of memory, to recover the full AK_7 and 32 bits of AK_0.

7 Conclusion

This paper described impossible differential distinguishers and attacks on reduced-round versions of Rijndael with blocks larger than 128 bits.

Table 1 compares the attack complexities of known attack on reduced-round Rijndael versions. The ID attacks presented in this paper do not threaten the full-version of any Rijndael cipher.

Table 1. Comparison of attacks on reduced-round versions of Rijndael

Cipher	#Rounds	Time	Data	Memory	Ref.	Comments
AES	5	2^{31}	$2^{29.5}$ CP	2^{32}	[6]	Imp. Diff.
	6	2^{122}	$2^{91.5}$ CP	2^{32}	[8]	Imp. Diff.
	7	2^{80}	2^{52} RK-CP	—	[27]	(ID) 2 rel. keys
	7	2^{94}	2^{56} RK-CP	—	[5]	(ID) 32 rel. keys
	7	2^{116}	2^{111} RK-CP	—	[13]	(ID) 2 rel. keys
	7	2^{145}	2^{37} RK-CP	—	[27]	(ID) 2 rel. keys
	7	2^{186}	2^{92} CP	—	[24]	Imp. Diff.
	8	2^{134}	2^{116} RK-CP	—	[5]	(ID) 32 rel. keys
	8	2^{136}	2^{112} RK-CP	—	[27]	(ID) 2 rel. keys
	8	2^{153}	2^{88} RK-CP	—	[27]	(ID) 2 rel. keys
	8	2^{159}	2^{92} RK-CP	—	[5]	(ID) 32 rel. keys
	8	2^{177}	$2^{64.5}$ RK-CP	—	[27]	(ID) 2 rel. keys
	8	2^{183}	2^{88} RK-CP	—	[13]	(ID) 2 rel. keys
	8	2^{184}	$2^{68.5}$ RK-CP	—	[5]	(ID) 32 rel. keys
Rijndael-160	4	2^{15}	2^{9} CP	2^{8}	[21]	Multiset
	5	2^{38}	2^{33} CP	2^{32}	[21]	Multiset
	5	2^{44}	$2^{10.5}$ CP	2^{8}	[21]	Multiset
	5	2^{61}	$2^{34.3}$ CP	2^{32}	Sect.3.1	Imp. Diff.
	6	$2^{43.5}$	$2^{34.5}$ CP	2^{32}	[21]	Multiset
	6	2^{135}	$2^{105.5}$ CP	$2^{69.7}$	Sect.3.2	Imp. Diff.
	7	$2^{133.5}$	2^{129} CP	2^{128}	[21]	Multiset
Rijndael-192	4	2^{14}	2^{9} CP	2^{8}	[21]	Multiset
	5	2^{25}	2^{32} CP	2^{32}	[21]	Multiset
	5	2^{61}	$2^{34.6}$ CP	2^{32}	Sect.4.1	Imp. Diff.
	6	$2^{43.5}$	2^{34} CP	2^{32}	[21]	Multiset
	6	2^{151}	$2^{121.5}$ CP	$2^{85.4}$	Sect.4.2	Imp. Diff.
	7	2^{141}	$2^{130.5}$ CP	2^{128}	[21]	Multiset
Rijndael-224	4	3	2^{8} CP	2^{8}	[21]	Multiset
	5	2^{36}	2^{10} CP	2^{8}	[21]	Multiset
	6	$2^{43.5}$	$2^{34.5}$ CP	2^{32}	[21]	Multiset
	6	2^{61}	$2^{34.8}$ CP	2^{32}	Sect.5.1	Imp. Diff.
	7	2^{141}	$2^{130.5}$ CP	2^{128}	[21]	Multiset
	7	2^{167}	2^{138} CP	2^{104}	Sect.5.2	Imp. Diff.
Rijndael-256	4	—	2^{8} CP	2^{8}	[21]	Multiset
	5	2^{38}	2^{33} CP	2^{32}	[21]	Multiset
	6	$2^{43.5}$	2^{33} CP	2^{32}	[21]	Multiset
	6	$2^{61.4}$	2^{35} CP	2^{32}	Sect.6.1	Imp. Diff.
	7	2^{141}	$2^{130.5}$ CP	2^{128}	[21]	Multiset
	7	2^{182}	2^{153} CP	2^{117}	Sect.6.2	Imp. Diff.

CP: Chosen-Plaintext; RK-CP: Related-Key Chosen-Plaintext

The 5-round ID distinguishers for Rijndael-224 and Rijndael-256 do not apply to the AES. These new results are due to the larger text blocks and the consequent slower diffusion (SR and MC) in the former.

The ID attack results do not depend on any particular key sets or subkey values (weak-key assumptions). Moreover, these distinguishers also hold for duals of Rijndael [2] since changing the irreducible polynomial, or the coefficients of MixColumns, or the S-box (as long as it is invertible) do not affect the distinguishers. We have not applied related-key impossible differentials on large-block Rijndael versions, such as [27], because key schedule algorithms have only been officially defined for the AES (128-bit text block), and not for larger block sizes of Rijndael [25].

References

1. AES The Advanced Encryption Standard Development Process (1997), http://csrc.nist.gov/encryption/aes/
2. Barkan, E., Biham, E.: In How Many Ways Can You Write Rijndael. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 160–175. Springer, Heidelberg (2002)
3. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials, Technion, CS Dept, Tech Report CS0947 (1998)
4. Biham, E., Biryukov, A., Shamir, A.: Miss-in-the-Middle Attacks on IDEA, Khufu and Khafre. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999)
5. Biham, E., Dunkelman, O., Keller, N.: Related-Key Impossible Differential Attacks on 8-round AES-192. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 21–33. Springer, Heidelberg (2006)
6. Biham, E., Keller, N.: Cryptanalysis of Reduced Variants of Rijndael, 3rd AES Conference, New York, USA (2000), http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html
7. Biryukov, A.: The Boomerang Attack on 5 and 6-round Reduced AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) Advanced Encryption Standard – AES. LNCS, vol. 3373, pp. 11–15. Springer, Heidelberg (2005)
8. Cheon, J.H., Kim, M., Kim, K., Lee, J.-Y., Kang, S.: Improved Impossible Differential Cryptanalysis of Rijndael and Crypton. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 39–49. Springer, Heidelberg (2002)
9. Daemen, J., Rijmen, V.: AES Proposal: Rijndael, 1st AES Conference, California, USA (1998), http://www.nist.gov/aes
10. Daemen, J., Rijmen, V.: The Design of Rijndael - AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
11. FIPS180-2, Secure Hash Standard (SHS), FIPS PUB 180-2, Federal Information Processing Standard Publication 180-2, National Institute of Standards and Technology (NIST) (August 2002)
12. FIPS197, Advanced Encryption Standard (AES), FIPS PUB 197 Federal Information Processing Standard Publication 197, U.S. Department of Commerce (November 2001)
13. Jakimoski, G., Desmedt, Y.: Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2003)
14. Jakobsen, T., Knudsen, L.R.: The Interpolation Attack on Block Ciphers. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 28–40. Springer, Heidelberg (1997)

15. Knudsen, L.R.: DEAL – a 128-bit Block Cipher, Technical Report #151, University of Bergen, Dept. of Informatics, Norway (February 1998)
16. Knudsen, L.R., Berson, T.A.: Truncated Differentials of SAFER. In: Gollmann, D. (ed.) Fast Software Encryption. LNCS, vol. 1039, pp. 15–26. Springer, Heidelberg (1996)
17. Langford, S.K.: Differential-Linear Cryptanalysis and Threshold Signatures, PhD thesis, Stanford University, USA (1995)
18. Lenstra, H.W.: Rijndael for Algebraists (April 2002), http://math.berkeley.edu/hwl/papers/rijndael0.pdf
19. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
20. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton, USA (1996)
21. Nakahara Jr., J., de Freitas, D.S., Phan, R.C.W.: New Multiset Attacks on Rijndael with Large Blocks. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 277–295. Springer, Heidelberg (2005)
22. NESSIE, New European Schemes for Signatures, Integrity and Encryption (January 2000), http://www.cryptonessie.org
23. NESSIE Deliverable D21, Performance of Optimized Implementations of the NESSIE Primitives, version 2.0 (February 20, 2003), https://www.cosic.esat.kuleuven.be/nessie/deliverables/
24. Phan, R.C.W.: Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES). Information Processing Letters 91(1), 33–38 (2004)
25. Rijmen, V.: private communication (2006)
26. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell System Technical Journal 28, 656–715 (1949)
27. Zhang, W., Wu, W., Zhang, L., Feng, D.: Improved Related-Key Impossible Differential Attacks on Reduced-Round AES-192. In: Biham, E., Youssef, A.M. (eds.) 13th Selected Areas in Cryptography, SAC 2006, pp. 101–118. Springer, Heidelberg (2006)

A Appendix A

We have found the 6-round ID distinguisher (6), which is longer than (4), but is ineffective for a key-recovery attack on Rijndael-224 because there are too many (24) zero output byte differences. The main novelty in this distinguisher is the Δ truncated differential which starts with four nonzero byte differences, δ, at AK_6 and evolves until AK_5 with four δ bytes in the leftmost column. After MC^{-1} of the fifth round, there can be up to four nonzero byte differences ('$*$') in that column. At least one of those '$*$' is nonzero, as guaranteed by the branch number of the MC matrix. Continuing with the difference propagation backwards, we arrive at the third round, where several contradictions happen at the columns in MC. For instance, before the MC layer, the leftmost column contains the values $(\delta, 0, 0, 0)$, and after the MC layer the same column contains $(*, 0, 0, 0)$. The sum of δs and '*'s before and after the MC layer is two. Whether the '*' is a zero or nonzero difference contradicts the branch number of Rijndael, which is five. A similar reasoning applies to (7), a 6-round ID distinguisher for Rijndael-192.

$$\begin{pmatrix} \delta&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} \overset{MC\circ SR\circ SB\circ AK_0}{\rightarrow} \begin{pmatrix} \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0 \end{pmatrix} \overset{SR\circ SB\circ AK_1}{\rightarrow} \begin{pmatrix} \delta&0&0&0&0&0&0\\ 0&0&0&0&0&0&\delta\\ 0&0&0&0&0&\delta&0\\ 0&0&0&\delta&0&0&0 \end{pmatrix} \overset{MC}{\rightarrow}$$

$$\begin{pmatrix} \delta&0&0&\delta&0&\delta&\delta\\ \delta&0&0&\delta&0&\delta&\delta\\ \delta&0&0&\delta&0&\delta&\delta\\ \delta&0&0&\delta&0&\delta&\delta \end{pmatrix} \overset{SR\circ SB\circ AK_2}{\rightarrow} \begin{pmatrix} \delta&0&0&\delta&0&\delta&\delta\\ 0&0&\delta&0&\delta&\delta&\delta\\ 0&\delta&0&\delta&\delta&\delta&0\\ 0&\delta&\delta&\delta&0&0&\delta \end{pmatrix} \overset{MC}{\not\rightarrow} \begin{pmatrix} *&*&*&0&*&0&0\\ 0&*&*&*&0&*&0\\ 0&0&*&*&*&0&*\\ 0&*&0&0&*&*&* \end{pmatrix} \overset{AK_3^{-1}\circ SB^{-1}\circ SR^{-1}}{\leftarrow}$$

$$\begin{pmatrix} *&*&*&0&*&0&0\\ *&*&*&0&*&0&0\\ *&*&*&0&*&0&0\\ *&*&*&0&*&0&0 \end{pmatrix} \overset{MC^{-1}}{\leftarrow} \begin{pmatrix} *&0&0&0&0&0&0\\ 0&*&0&0&0&0&0\\ 0&0&*&0&0&0&0\\ 0&0&0&0&*&0&0 \end{pmatrix} \overset{AK_4^{-1}\circ SB^{-1}\circ SR^{-1}}{\leftarrow}$$

$$\begin{pmatrix} *&0&0&0&0&0&0\\ *&0&0&0&0&0&0\\ *&0&0&0&0&0&0\\ *&0&0&0&0&0&0 \end{pmatrix} \overset{MC^{-1}\circ AK_5^{-1}\circ SB^{-1}}{\leftarrow} \begin{pmatrix} \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0 \end{pmatrix} \overset{SR^{-1}\circ AK_6^{-1}}{\leftarrow} \begin{pmatrix} \delta&0&0&0&0&0&0\\ 0&0&0&0&0&0&\delta\\ 0&0&0&0&0&\delta&0\\ 0&0&0&\delta&0&0&0 \end{pmatrix} \quad (6)$$

$$\begin{pmatrix} \delta&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} \overset{MC\circ SR\circ SB\circ AK_0}{\rightarrow} \begin{pmatrix} \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0\\ \delta&0&0&0&0&0&0 \end{pmatrix} \overset{SR\circ SB\circ AK_1}{\rightarrow} \begin{pmatrix} \delta&0&0&0&0&0&0\\ 0&0&0&0&0&0&\delta\\ 0&0&0&0&0&\delta&0\\ 0&0&0&\delta&0&0&0 \end{pmatrix} \overset{MC}{\rightarrow}$$

$$\begin{pmatrix} \delta&0&0&\delta&\delta&\delta&\delta\\ \delta&0&0&\delta&\delta&\delta&\delta\\ \delta&0&0&\delta&\delta&\delta&\delta\\ \delta&0&0&\delta&\delta&\delta&\delta \end{pmatrix} \overset{SR\circ SB\circ AK_2}{\rightarrow} \begin{pmatrix} \delta&0&0&\delta&\delta&\delta&\delta\\ 0&0&\delta&\delta&\delta&\delta&\delta\\ 0&\delta&\delta&\delta&\delta&\delta&0\\ \delta&\delta&\delta&\delta&0&0&\delta \end{pmatrix} \overset{MC}{\not\rightarrow} \begin{pmatrix} *&*&*&*&*&0&0\\ 0&*&*&*&*&*&0\\ 0&0&*&*&*&*&*\\ *&0&0&*&*&* \end{pmatrix}$$

$$\overset{AK_3^{-1}\circ SB^{-1}\circ SR^{-1}}{\leftarrow} \begin{pmatrix} *&*&*&*&0&0\\ *&*&*&*&0&0\\ *&*&*&*&0&0\\ *&*&*&*&0&0 \end{pmatrix} \overset{MC^{-1}}{\leftarrow} \begin{pmatrix} *&0&0&0&0&0\\ 0&*&0&0&0&0\\ 0&0&*&0&0&0\\ 0&0&0&*&0&0 \end{pmatrix} \overset{AK_4^{-1}\circ SB^{-1}\circ SR^{-1}}{\leftarrow}$$

$$\begin{pmatrix} *&0&0&0&0&0\\ *&0&0&0&0&0\\ *&0&0&0&0&0\\ *&0&0&0&0&0 \end{pmatrix} \overset{MC^{-1}\circ AK_5^{-1}\circ SB^{-1}}{\leftarrow} \begin{pmatrix} \delta&0&0&0&0&0\\ \delta&0&0&0&0&0\\ \delta&0&0&0&0&0\\ \delta&0&0&0&0&0 \end{pmatrix} \overset{SR^{-1}\circ AK_6^{-1}}{\leftarrow} \begin{pmatrix} \delta&0&0&0&0&0\\ 0&0&0&0&0&\delta\\ 0&0&0&0&\delta&0\\ 0&0&0&\delta&0&0 \end{pmatrix} \quad (7)$$

High-Speed Pipelined Hardware Architecture for Galois Counter Mode

Akashi Satoh[1], Takeshi Sugawara[2], and Takafumi Aoki[2]

[1] National Institute of Advanced Industrial Science and Technology,
1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan
akashi.satoh@aist.go.jp
[2] Graduate School of Information Sciences, Tohoku University
Sendai, Miyagi, Japan
sugawara@aoki.ecei.tohoku.ac.jp

Abstract. In the authenticated encryption mode GCM (Galois Counter Mode), the CTR (counter) mode for data encryption that has no feedback path can easily be pipelined to boost the operating frequency of a hardware implementation. However, the hash function for the authentication tag generation performs multiply-add operations sequentially by chaining the result in the previous cycle, and this becomes the critical path in the high-speed GCM hardware. Therefore, we propose a high-speed pipelined hardware architecture for GCM in conjunction with a pipelined multiply-adder on a Galois field GF(2^{128}). This architecture was implemented with a 4-stage pipelined multiply-adder and a 56-stage pipelined AES (Advanced Encryption Standard) circuit by using a 0.13-um CMOS standard cell library. This implementation showed very high throughput of 54.94 Gbps with 272 Kgates for the key lengths of 128, 192, and 256 bits. The high hardware efficiency (throughput/gate) of 201.75 Kbps/gate is also an improvement over prior art.

1 Introduction

GCM [1] is an authenticated encryption mode that generates cipher text and an authentication tag simultaneously. The CTR (counter) mode [2] is used for the encryption, and a hash function repeats multiply-add operations over GF(2^{128}) to generate the tag. The National Institute of Standards and Technology (NIST) standardized GCM as SP 800-38D [3]. GCM with AES [4] (GCM-AES) has also been adopted in many standards such as RFC 4106 [5] for payload encryption in IPSec (by the Internet Engineering Task Force (IETF)), IEEE 802.1AE [6] for frame data encryption in the Ethernet protocol, and IEEE P1619.1 [7] for tape storage encryption.

We have developed high performance GCM hardware architectures [9] [10], which have a good scalability between circuit size and throughput, and evaluated their performances in combination with various AES circuits. The single 128-bit multiply-add operation on GF(2^{128}) for the hash function is much faster than the single 128-bit block encryption with AES, but it is easy to boost the throughput of the encryption by utilizing the parallel operating capability of the CTR mode with multiple AES hardware cores. In contrast, the multiply-add operation is intrinsically sequential, and thus,

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 118–129, 2007.

the GF(2^{128}) multiplier becomes the critical path when the AES part is made highly parallel. In reference [9], we proposed a parallel multiplier architecture for GCM hardware to solve this problem. However, the architecture needs an additional 256-bit I/O port (128 bits for plaintext and 128 bits for ciphertext) for each additional parallel processing block that contains one multiply-adder and one AES circuit. For example, the 4-parallel version in the reference has a 512-bit data input port and a 512-bit data output port.

In this paper, we propose a high-speed GCM hardware architecture where the critical path of the multiplier is divided into multiple pipeline stages to boost operating frequency. The input data runs through the single pipelined data path, and thus we do not need to increase the number of I/O ports.

This paper is organized as follows. In Section 2, a pipelined multiply-adder for the hash function is presented with a detailed example of a 4-stage version. The datapath architecture of the GCM hardware along with a pipelined AES circuit using a 3-stage composite field S-box is described in Section 3. Section 4 evaluates its performance by using a CMOS ASIC library in a comparison with prior art. Finally, the conclusions are described in Section 4.

2 Pipelined Multiply-Adder for Hash Function

The hash function of GCM repeats multiply-add operations according to Equation (A.3) in the Appendix. Reference [1] showed 2-parallel multiply-add operations for the hash function, where the input data is interleaved for $\{A_1, A_3, A_5, \cdots\}$ and $\{A_2, A_4, A_6, \cdots\}$ as in Equation (1).

$$\begin{aligned} X_i &= (X_{i-1} \oplus A_i)H \\ &= (\cdots(((((A_1H \oplus A_2)H \oplus A_3)H \oplus A_4)H \oplus A_5)H \oplus A_6)H \cdots)H \\ &= ((A_1H^2 \oplus A_3)H^2 \oplus A_5 \cdots)H^2 \\ &\quad \oplus ((A_2H^2 \oplus A_4)H^2 \oplus A_6 \cdots)H \end{aligned} \tag{1}$$

Not only 2-parallelism, but higher parallelism is possible, such as 4-parallel operation where four consecutive inputs (for example A_1, A_2, A_3, and A_4) are processed at the same time as shown in the following Equation.

$$\begin{aligned} X_i = &(((A_1H^4 \oplus A_5)H^4 \oplus A_9)H^4 \oplus \cdots)H^4 \\ &\oplus(((A_2H^4 \oplus A_6)H^4 \oplus A_{10})H^4 \oplus \cdots)H^3 \\ &\oplus(((A_3H^4 \oplus A_7)H^4 \oplus A_{11})H^4 \oplus \cdots)H^2 \\ &\oplus(((A_4H^4 \oplus A_8)H^4 \oplus A_{12})H^4 \oplus \cdots)H \end{aligned} \tag{2}$$

We can generalize this equation to q-parallel processing as Equation (3), where the number of data block is pq. The q GF(2^{128}) multiply-adders are used for this equation,

and q data blocks $\{A_1, A_2, \cdots, A_q\}$ are processed in the first cycle, then $\{A_{1+q}, A_{2+q}, \cdots, A_{2q}\}$ are done in the second cycle, and so on.

$$\begin{aligned} X_i = &(\cdots((A_1H^q \oplus A_{q+1})H^q \oplus A_{2q+1})H^q \oplus \cdots \oplus A_{(p-1)q+1})H^q \\ &\oplus(\cdots((A_2H^q \oplus A_{q+2})H^q \oplus A_{2q+2})H^q \oplus \cdots \oplus A_{(p-1)q+2})H^{q-1} \\ &\quad \cdots \\ &\oplus(\cdots((A_qH^q \oplus A_{2q})H^q \oplus A_{3q})H^q \oplus \cdots \oplus A_{pq})H \end{aligned} \tag{3}$$

Fig. 1 shows the 4-parallel hardware architecture for Equation (2) proposed in reference [10], which uses four 128-bit GF multipliers. The AES circuit needs to feed four 128-bit ciphertext data blocks to the multiply-adder every operating cycle, and thus four 128-bit plaintext input and four 128-bit ciphertext ports are required for the GCM circuit using this 4-parallel architecture.

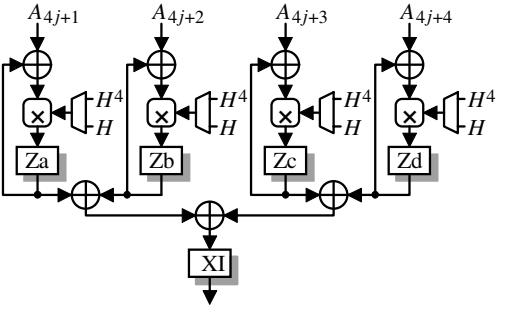


Fig. 1. Parallel architecture for multiply-addition (q=4)

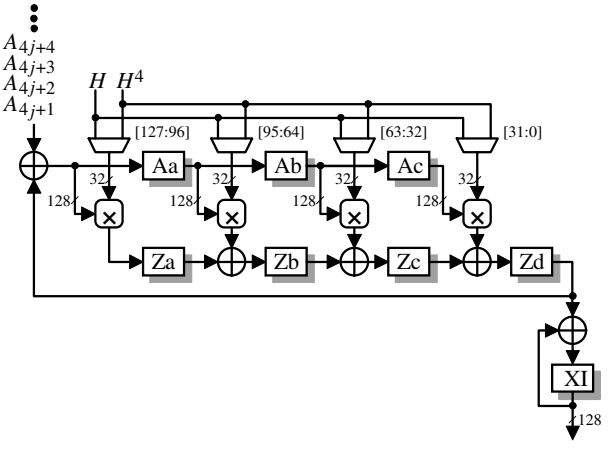


Fig. 2. Proposed pipelined architecture for multiply-addition (q=4)

Fig. 2 shows the pipelined multiply-adder architecture proposed in the current paper. This circuit is a 4-stage pipelined version based on the same Equation (2) used in Fig. 1. A 128-bit × 128-bit multiplier is divided into four 128-bit × 32-bit sub-multipliers, and three 128-bit registers Za, Zb, and Zc are inserted to shorten the critical path. Therefore one 128-bit multiplication is executed in four clock cycles, but the throughput remains at 128-bits/clock because four independent multiplications are processed in each sub-multiplier. This architecture can receive a 128-bit input every clock cycle for any number of pipeline stages. Therefore, the operating frequency can easily be boosted by increasing the number of pipeline stages.

Fig. 3 is an example operation for the 4-stage pipelined multiply-adder with 5 data blocks A_1~A_5, and is explained with Equations (4.a)~(4.o). Here, we assume that the value H^4 has already been calculated, and the number of input data blocks is given before the multiply-add operations of the hash function start. In this example, Ha~Hd and Ha^4~Hd^4 represent bits [127:96], [95:64], [63:32], and [31:0] of H and H^4, respectively.

$$Za_1 = A_1 Ha^4 \tag{4.a}$$

$$Za_2 = A_2 Ha^4,\ Zb_1 = Za_1 \oplus A_1 Hb^4 \tag{4.b}$$

$$\begin{cases} Za_3 = A_3 Ha,\ Zb_2 = Za_2 \oplus A_2 Hb^4 \\ Zc_1 = Zb_1 \oplus A_1 Hc^4 \end{cases} \tag{4.c}$$

$$\begin{cases} Za_4 = A_4 Ha,\ Zb_3 = Za_3 \oplus A_3 Hb \\ Zc_2 = Zb_2 \oplus A_2 Hc^4,\ Zd_1 = Zc_1 \oplus A_1 Hd^4 \end{cases} \tag{4.d}$$

$$\begin{cases} A_5^{'} = A_5 \oplus Zd_1 \\ Za_5 = A_5^{'} Ha,\ Zb_4 = Za_4 \oplus A_4 Hb \\ Zc_3 = Zb_3 \oplus A_3 Hc,\ Zd_2 = Zc_2 \oplus A_2 Hd^4 \end{cases} \tag{4.e}$$

$$\begin{cases} Zb_5 = Za_4 \oplus A_5^{'} Hb,\ Zc_4 = Zb_4 \oplus A_4 Hc \\ Zd_3 = Zc_3 \oplus A_3 Hd \\ X_1 = Zd_2 \end{cases} \tag{4.f}$$

$$\begin{cases} Za_6 = Zd_3 Ha,\ Zc_5 = Zb_5 \oplus A_5^{'} Hc \\ Zd_4 = Zc_4 \oplus A_4 Hd \end{cases} \tag{4.g}$$

$$\begin{cases} Za_7 = Zd_4 Ha,\ Zb_6 = Za_6 \oplus Zd_3 Hb \\ Zd_5 = Zc_5 \oplus A_5^{'} Hd \end{cases} \tag{4.h}$$

$$\begin{cases} Zb_7 = Za_7 \oplus Zd_4 Hb,\ Zc_6 = Zb_6 \oplus Zd_3 Hc \\ X_2 = X_1 \oplus Zd_5 \end{cases} \tag{4.i}$$

$$Zc_7 = Zb_7 \oplus Zd_4 Hc,\ Zd_6 = Zc_6 \oplus Zd_3 Hd \tag{4.j}$$

$$Za_8 = Zd_6 Ha,\ X_3 = X_2 \oplus Zd_7 \tag{4.k}$$

$$Zb_8 = Za_8 \oplus Zd_6 Hb,\ X_3 = X_2 \oplus Zd_7 \tag{4.l}$$

$$Zc_8 = Zb_8 \oplus Zd_6 Hc \tag{4.m}$$

$$Zd_8 = Zc_8 \oplus Zd_6 Hd \tag{4.n}$$

$$X_4 = X_3 \oplus Zd_8 \tag{4.o}$$

In Fig. 3(a), after all of the 128-bit registers are cleared to 0, the 128-bit input A_1 is multiplied by Ha^4 (the most significant 32 bits of H^4), and the 128-bit result Za_1 is stored into the register Za in the next cycle. At the same time, A_1 is stored into the register Aa. In Fig. 3(b), A_1 is multiplied by Hb^4 (bits 95~64 of H^4) and added to Za_1, and the result Zb_1 is stored into the register Zb. Then A_1 is moved from the register Aa to Ab. In Figs. 3(c)~(d), A_1Hc^4 and A_1Hd^4 are calculated and summed up one after another, and finally, the 128-bit product of the input A_1 and the constant value H^4 is obtained in the register Zd. In Fig. 3(b), the second block A_2 is input and is processed in parallel with the multiply-add operation on A_1. In Figs. 3(c) and (d), A_3 and A_4 are input to be processed, respectively. Therefore, four multiply-add (128-bit × 32-bit + 128-bit) operations are processed simultaneously in this example.

Note that A_3 and A_4 need to be multiplied by H^3 and H^2, respectively. Therefore, they are multiplied by H three times and twice, respectively, by rotating them in the pipeline loop. H^3 and H^2 can be calculated in advance, but that is of no net benefit because extra clock cycles are required to calculate the values, and extra registers are needed to hold them. In summary, A_1, A_2, A_3, A_4, and A_5 are multiplied by H^5(=H^4×H), H^4, H^3(=H×H×H), H^2(=H×H), and H, respectively.

In Fig. 3(e), the final input block A_5 is added to Zd_5, and then the operation to calculate following value is started.

$$(A_5 \oplus Zd_1)H = (A_5 \oplus A_1 H^4)H = A_5 H \oplus A_1 H^5 \tag{5}$$

The 128-bit temporary value $A_5' = A_5 \oplus Zd_1$ is shifted thorough registers Aa~Ac, and is sequentially multiplied by the 32-bit constants Ha, Hb, Hc, and Hd during Figs. 3(e)~(h). Then the partial products are summed up using the registers Aa~Zd. In parallel, the following operations are done. In Fig. 3(f), the value Zd_2 (=A_2H^4) held in the register Zd is transferred to the register XI as X_1. In Fig. 3(g), Zd_3 (=A_3H) is fed back to the register Aa to be multiplied by H two more times, and Zd_3×Ha is also fed back to the register Za. In this cycle, the registers Aa and Za are not used, because Zd_2 (=A_2H^4) has not been fed back to the registers but has been stored in XI. In Fig. 3(h), Zd_4 (=A_4H) is also fed back to the registers to be multiplied by H.

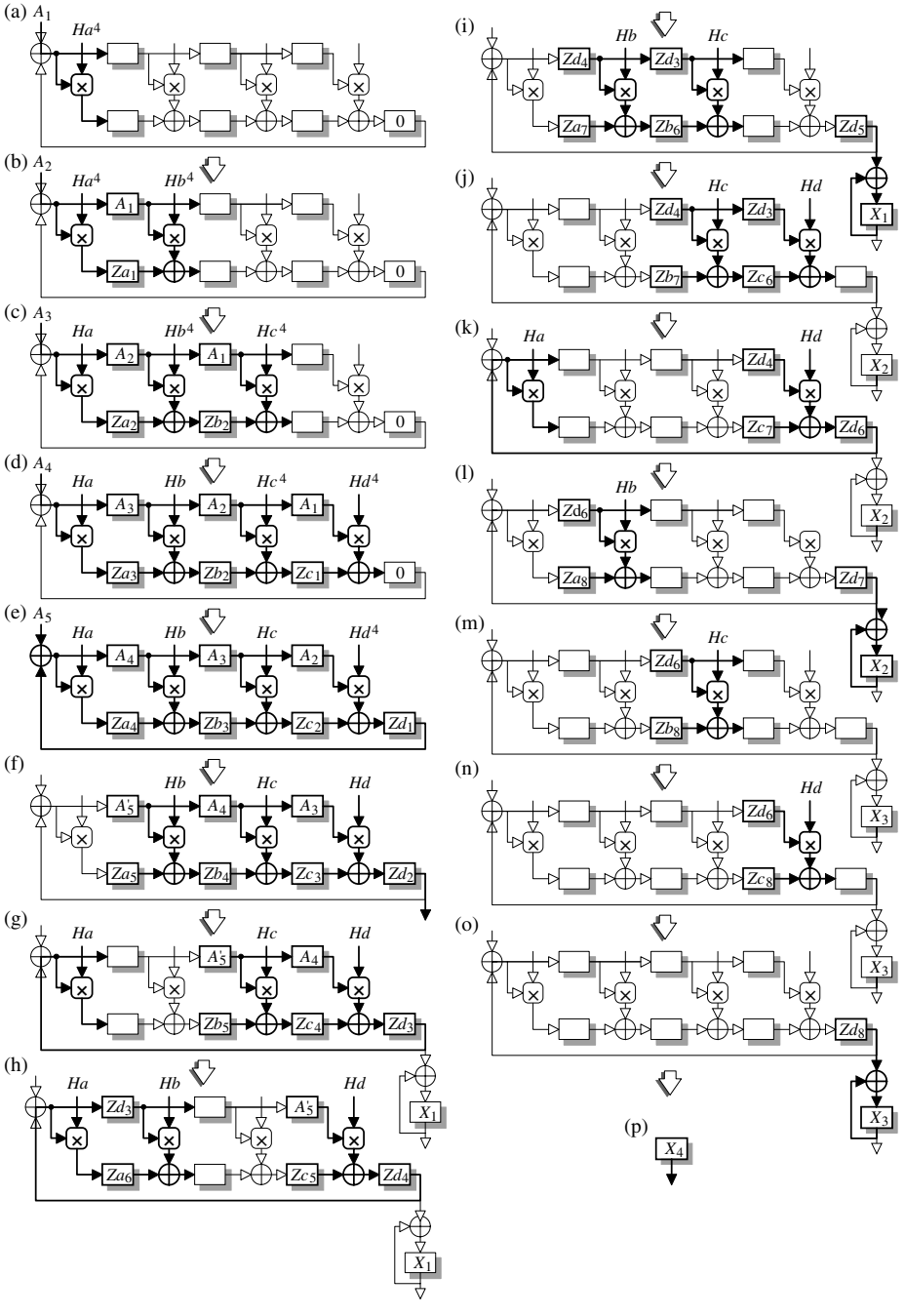


Fig. 3. Example of pipelined operations for Equations (4.a)~(4.o)

In Fig 3(i), $Zd_5 (= A_5H \oplus A_1H^4)$ from the register Zd is added to X_1 in the register XI, and X_2 is obtained.

$$X_2 = X_1 \oplus Zd_5 = A_1H^5 \oplus A_2H^4 \oplus A_5H \tag{6}$$

In Figs. 3(j)~(k), Zd_6 (=A_3H^2) is obtained in the register Zd, but A_3 should be multiplied by H^3, and thus Zd_6 is not summed into the register XI, but is fed back to the registers Aa and Za. In contrast, Zd_7 (=A_4H^2) is summed into the register XI in Fig. 3(l) and the value X_3 is obtained in XI.

$$X_3 = X_2 \oplus Zd_6 = A_1H^5 \oplus A_2H^4 \oplus A_4H^2 \oplus A_5H \tag{7}$$

In Figs. 3(l)~(o), Zd_8 (=A_3H^3) is calculated in the register Zd, and then the final result X_4 of Equation (8) is obtained by adding Zd_8 to X_3 in the register XI.

$$\begin{aligned} X_4 &= X_3 \oplus Zd_7 \\ &= A_1H^5 \oplus A_2H^4 \oplus A_3H^3 \oplus A_4H^2 \oplus A_5H \end{aligned} \tag{8}$$

3 Pipelined GCM Hardware Architecture

Fig. 4 shows the architecture of the GCM circuit, where the 4-stage pipelined multiply-adder described in the previous section is used for the hash function. The register Z is added to the feedback path in the hash function block of Fig. 4, which is not included in Fig. 2. This is only to shorten the critical path but the basic operation is not

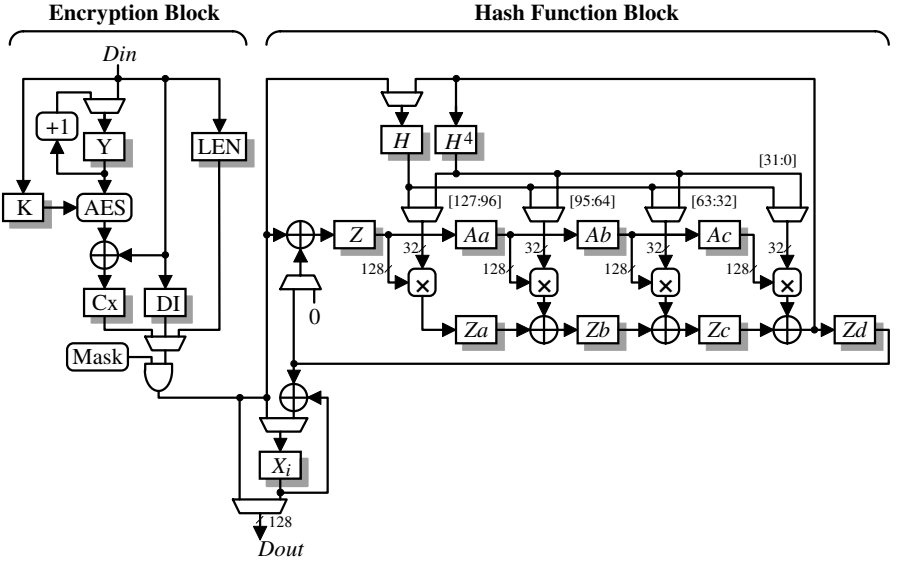


Fig. 4. Proposed GCM hardware architecture (q=4)

changed. In our previous papers [9] and [10], we implemented several GCM-AES circuits with the composite field [12] and BDD S-boxes for compact and high-speed implementations, respectively. However, only the composite field S-box was implemented this time, because the inside of the composite field S-box can easily be pipelined to increase the operating frequency, as shown in Fig. 5. The fast but large BDD S-box is needed to execute the S-box operation in single clock cycle.

The AES encryption circuit has an unrolled architecture with 14 round function blocks, and supports 128-, 192-, and 256-bit keys. Each round function block has the 4-stage (3 stages for S-box and 1 stage for the rest) pipelined structure. Therefore, the latencies for 128-, 192-, and 256-bit keys are 4×10=40 clocks, 4×12=48 clocks, and 4×14=56 clocks, respectively. When data blocks are fed continuously, a maximum throughput of 128 bits/clock is obtained. Both the round function block and the multiply-adder have 4-pipline stages, but these fortuitously happened to coincide when we had balanced the critical path delays in the AES and the hash function blocks. All the round keys are pre-calculated by a key scheduler not shown in the figures and stored in key registers. The register to register feedback path of the key scheduler contains an S-box, several XORs, and some selectors, which made it longer than the round function, so the path was divided into 5 stages.

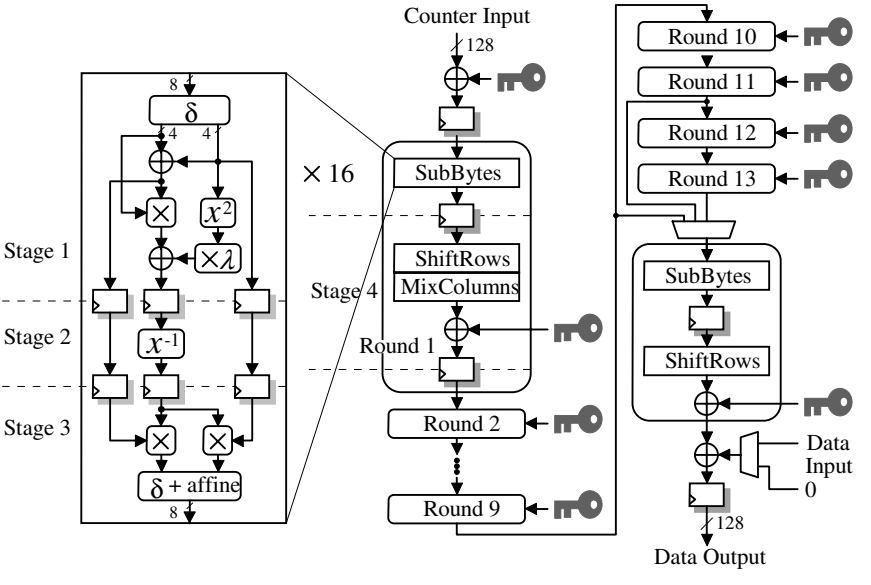


Fig. 5. Pipelined AES circuit (q=4)

4 Hardware Performance Comparison in ASICs

The new AES-GSM circuit was synthesized by using a 0.13-μm CMOS standard cell library [15], and the results are shown in Table 1 in comparison with prior art. The

Table 1. Performance Comparison of GCM

Ref.	GCM Architecture	AES Architecture	# Round Function Blocks	Key Length (bits)	S-Box	Critical Path (ns)	Optimize	Max. Freq. (MHz)	Thr'put (Gbps)	Size (gates)	Kbps /gate	Library (μm)
This Work	Pipelined	4-Innner Pipelined	14	128, 192, 256	Composite	3.33	Size	300.3	38.44	262,118	146.65	0.13
						2.33	Speed	429.2	54.94	272,291	201.75	
[9]	Sequential	Pipelined	14		Composite	5.00	Size	200.0	25.60	174,016	147.11	
						4.00	Speed	250.0	32.00	181,198	176.60	
					BDD	3.00		333.3	42.67	297,542	143.40	
		4-stage Pipelined Loop	4		Composite	5.00	Size	200.0	6.40	73,104	87.55	
						4.00	Speed	250.0	8.00	79,566	100.55	
					BDD	3.00		333.3	10.67	118,645	89.90	
	4 Parallel	Loop	4		Composite	5.00	Size	200.0	6.40	96,241	66.50	
						4.00	Speed	250.0	8.00	106,893	74.84	
					BDD	3.00		333.3	10.67	162,373	65.69	
[10]	4 Parallel	Pipelined	14×4		Composite	5.00	Size	200.0	102.40	600,440	170.54	
						4.00	Speed	250.0	128.00	697,567	183.49	
					BDD	3.15		317.5	162.56	979,348	165.99	
[12]						3.69		271.0	34.69	498.658	69.57	0.18
[13]						3.33		300.0	7.00	97,000	72.1	
				128					10.00	190,000	52.6	
				128					20.00 40.00	180,000 330,000	111.1 121.2	
[14]	Sequential	Loop	1	128		4.00		250	3.2	30,707	104.2	0.13
				256		2.00		500	6.4	40,335	158.6	
						1.21		824	10.5	49,633	211.5	0.09

AES datapath of the proposed architecture is similar to that of "Sequential GCM + Pipelined AES" in Reference [9] that has 14 round function blocks, but the round function blocks and the multiply-adder are divided into several stages to shorten the critical path. As a result, the operating frequency of the composite field S-box version was increased by 70% (from 250.0 MHz to 429.2 MHz). Our new design also achieved the 30% higher throughput of 54.94 Gbps with a smaller size of 272 Kgates in comparison to the BDD S-box version of reference [9], whose throughput is 42.67 Gbps with 298 Kgates. The hardware efficiency defined as throughput per gate is 201.75 Kbps/gate for the proposed architecture, which is the highest among all of the GCM hardware we have designed [9] [10].

The proposed architecture is much faster than the GCM hardware in other references [12][13][14], and only the one with a simple loop-architecture synthesized by using a 0.09-μm CMOS library in [14] has a higher hardware efficiency of 211.55 Kbps/gate (=10.5Gbps/49,633gates). This is only due to the natural speed advantage of the 0.09-μm process technology, and the throughput of the same design synthesized by using a 0.13-μm CMOS library is 158.6 Kbps/gate, which is 20% lower than ours. The throughput of the 0.09-μm implementation is calculated as 128 bits × 824 MHz / 10 clocks = 10.55 Gbps, and thus this value is for a 128-bit key that takes 10 clock cycles for one 128-bit block encryption. Therefore, when a 256-bit key that requires 14 clock cycles is used, the throughput would be degraded down to 128 bits × 824 MHz / 14 clocks = 7.53 Gbps, and the hardware efficiency would be 7.53 Gbps

/ 49,633 gates = 151.79 Kbps/gate. For the 0.13-μm implementation, these value would be 128bits × 500 MHz / 14 clocks = 4.57 Gbps and 4.57 Gbps / 40,335 gates = 113.34 Kbps/gates. Though a 192-bit key is not supported in [14], our implementation achieved a throughput of 54.9 Gbps and the hardware efficiency of 201.75 Kbps/gate for the 128-, 192-, and 256-bit keys. As a result, our proposed architecture has advantages over the prior art in terms of hardware efficiency.

5 Conclusion

We proposed a pipelined Galois field multiply-adder to boost the operating frequency of the hash function block that was the bottleneck for higher speed GCM hardware, and evaluated its performance using a 0.13-μm CMOS standard cell library. The GCM-AES hardware implemented with the 4-stage pipelined hash function block and the 56-stage AES block using the 3-stage composite field S-box achieved a very high throughput of 54.94 Gbps with 272 Kgates. Our design supports key lengths of 128, 192, and 256 bits, and the throughput is independent of the key size. Further speed-up is possible by increasing the number of pipeline stages in the critical path to achieve higher operating frequencies. In addition to the speed, the hardware efficiency defined as throughput per gate was also measured, showing the strong advantage of the proposed architecture over the prior art.

References

1. McGrew, D., et al.: The Galois/Counter Mode of Operation (GCM) (May 2005), http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-revised-spec.pdf
2. NIST, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, Special Publication 800-38A (December 2001), http://csrc.nist.gov/CryptoToolkit/modes/800-38_Series_Publications/SP800-38A.pdf
3. NIST, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication, Draft Special Publication 800-38D (April 2006), http://csrc.nist.gov/publications/drafts/Draft-NIST_SP800-38D_Public_Comment.pdf
4. NIST, Advanced Encryption Standard (AES) FIPS Publication 197 (November 2001), http://csrc.nist.gov/ publications/fips/fips197/ fips-197.pdf
5. Viega, J., et al.: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (EPS) RFC 4106 (June 2005), http://www.faqs.org/rfcs/rfc4106.htm
6. IEEE, 802.1AE - Media Access Control (MAC) Security, Draft 3.5 (June 2005), http://www.ieee802.org/1/pages/802.1ae.html
7. IEEE, P, 1/D12a - Standard for Authenticated Encryption with Length Expansion for Storage Devices (November 2006), http://grouper.ieee.org/groups/ 1619/ email/bin00084.bin
8. Kohno, T., et al.: Carter Wegman (authentication) with Counter (encryption) (May 2003), http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/cwc/cwc-spec.pdf
9. Satoh, A.: High-Speed Hardware Architectures for Authenticated Encryption Mode GCM. In: Proc. IEEE ISCAS 2006, IEEE Computer Society Press, Los Alamitos (2006)
10. Satoh, A.: High-Speed Parallel Hardware Architecture for Galois Counter Mode. In: IEEE ISCAS 2007, IEEE Computer Society Press, Los Alamitos (2007)

11. Satoh, A., et al.: A Compact Rijndael Hardware Architecture with S-box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
12. Yang, B., et al.: High Speed Architecture for Galois/Counter Mode of Operation (GCM), Cryptology ePrint Archive: Report 2005/146 (June 2005), http://eprint.iacr.org/2005/146.pdf
13. Elliptic Semiconductor Inc, CLP-15/-16/-24 AES-GCM Core Preliminary Data Sheet (2004), http://www.ellipticsemi.com/
14. IP Cores, Inc., GCM1/GCM2 802.1ae (MACSec) GCM/AES Cores, 2006. http://www.ipcores.com/IEEE802.1AE-AES-GCM-Core.htm
15. Cu-11, I.B.M.: Standard Cell / Gate Array ASIC, http://www-03.ibm.com/chips/products/asics/products/cu-11.html

Appendix: Galois Counter Mode

Fig. A shows an example of GCM operations. GCM receives a secret key K, an initial vector IV (96 bits are recommended) for the counter value Y_0, authenticated data A, and plain text P for encryption, and then generates cipher text C and an authentication tag T. The authenticated data A and the plain text P are divided into n and m 128-bit blocks ($A_1, A_2, \cdots A_{m-1}, A_m^*$ and $P_1, P_2, \cdots P_{n-1}, P_n^*$), respectively. When a final block A_m^* or P_n^* is shorter than 128 bits (u and v bits, respectively, are used in the following equations), the rest of the block is filled with 0s. The authenticated encryption operation is defined as follows.

$$
\begin{aligned}
&H = Enc(K, 0^{128}) \\
&Y_0 = \begin{cases} IV \,\|\, 0^{31}1 & \text{if len}(IV) = 96 \\ GHASH(H, \{\}, IV) & \text{otherwise} \end{cases} \\
&C_0 = Enc(K, Y_0) \\
&Y_i = Y_{i-1} + 1 \qquad\qquad i = 1, ..., n \\
&C_i = P_i \oplus Enc(K, Y_i) \qquad i = 1, ..., n \\
&C_n^* = MSB_u(P_n \oplus E(K, Y_i)) \\
&T = MSB_t(GHASH(H, A, C) \oplus C_0)
\end{aligned}
\tag{A1}
$$

The hash function $GHASH(\ )$ receives a 128-bit constant H determined by the key K, the m-block authenticated data A, and the n-block ciphertext C, and then produces a 128-bit hash value. The initial value IV is encrypted and then XORed with the hash value, and the first t bits of the result become the authentication tag T.

In decryption, an authentication tag T' is reproduced from the authenticated data A and the ciphertext C, and then T' is checked to match the correct tag T.

The Galois field GF(2^{128}) used in GCM is defined by the following irreducible polynomial.

$$g(x) = x^{128} + x^7 + x^2 + x + 1 \tag{A2}$$

The hash function $GHASH(\ )$ defined over the field repeats multiplication and addition as shown in Equation (A3).

$$X_i = \begin{cases} 0 & i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & i = 1, ..., m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{128-v})) \cdot H & i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & i = m+1, ..., m+n-1 \\ (X_{m+n-1} \oplus (C_n^* \| 0^{128-u})) \cdot H & i = m+n \\ (X_{m+n} \oplus (\text{len}(A) \| \text{len}(C))) \cdot H & i = m+n+1 \end{cases} \tag{A3}$$

Then the final value X_{m+n+1} becomes the hash value.

$$X_{m+n+1} = GHASH(H, A, C) \tag{A4}$$

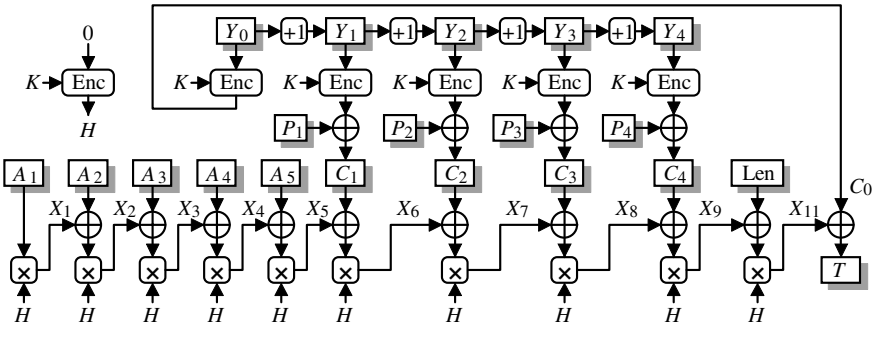


Fig. A. Example GCM operation (m = 5, n = 4)

Efficient Committed Oblivious Transfer of Bit Strings

Mehmet S. Kiraz, Berry Schoenmakers, and José Villegas

Dept. of Mathematics and Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
m.kiraz@tue.nl, berry@win.tue.nl, j.a.villegas@tue.nl

Abstract. Oblivious transfer (OT) is a powerful primitive in modern cryptography, often used in a context of semi-honest adversaries. Committed oblivious transfer (COT) is an enhancement involving the use of commitments, which can be used in many applications of OT covering particular malicious adversarial behavior. For OT, many protocols are known that cover the transfer of bit strings rather than just single bits. For COT, though, the known protocols only cover the transfer of bits.

In this paper, we thus present efficient COT protocols for transferring (long) bit strings, which perform quite well in comparison to the most efficient COT protocols for bits. We prove the security of our protocols following the simulation paradigm in the cryptographic model, also assuming the random oracle model for efficient non-interactive proofs. Also, as a motivation for the use of COT instead of OT, we point out that a protocol which uses OT as a subprotocol may have subtle security issues in the presence of malicious adversaries.

Keywords: Committed oblivious transfer, Commitments, Homomorphic encryption.

1 Introduction

Oblivious transfer is a fundamental primitive in modern cryptography. After Rabin introduced oblivious transfer [Rab81] a huge number of papers appeared regarding possible extensions, variants and applications of oblivious transfer. In Rabin's original oblivious transfer, the sender has a secret and sends it to a chooser who receives the secret with probability 1/2 while the sender does not know whether the secret has been received. Later, Even, Goldreich and Lempel [EGL85] presented 1-out-of-2 Oblivious Transfer (OT), where the sender has two values s_0 and s_1 and the chooser has a selection bit b. Upon completion of the protocol, the chooser holds the value s_b while the sender does not know which of the two values s_0 and s_1 the chooser got who, in turn, learns nothing about s_{1-b}. Crépeau [Cré87] showed that Rabin's OT and 1-out-of-2 OT are equivalent. 1-out-of-2 OT will be called standard OT throughout the rest of the paper.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 130–144, 2007.

Committed Oblivious Transfer (COT) is obtained as a natural combination of 1-out-of-2 oblivious transfer and bit commitments. This notion was first introduced by Crépeau [Cré90] under the name Verifiable Oblivious Transfer. Later, Crépeau *et al.* [CvdGT95] presented a more efficient COT protocol and showed that from COT one can construct a protocol for general secure multi-party computation in the malicious case. Briefly, in these variants of oblivious transfer, the parties running the protocol are committed to their input values prior to the protocol. That is, if the sender has two private values to be obliviously transferred to the chooser who has a selection bit, all these input values must have been committed to by the respective parties before the transfer starts. At the end of the protocol, the chooser will receive one of the corresponding private values of the sender together with a (public) commitment by the chooser.

For many applications in which OT is used as a subprotocol, the security of the overall protocol is considered only in the semi-honest model. However, there may be subtle security issues when such protocols using OT are extended to the malicious case. We highlight this in this paper and we describe how COT helps to overcome these problems. Namely, the link between OT and the surrounding protocol can be securely done with the use of COT.

Since efficient COT protocols for transferring bits are known, one may thus replace applications of OT of bits by COT of bits. An interesting question is how to do the same when transferring bit strings. This is the starting point of this work. In this paper, we present *efficient* protocols for string COT based on any (2,2)-threshold homomorphic cryptosystem.

Related Work

Garay *et al.* [GMY04] present the most efficient COT protocol to date realizing COT functionality in the Universal Composable (UC) framework of Canetti [Can00]. However, this protocol only works for bits whereas our protocol allow for bit strings of arbitrary length (up to the length of plaintexts of the underlying threshold homomorphic cryptosystem, or a multiple thereof). In this paper, we will just consider a stand-alone setting, noting that the efficiency of our protocols improves the efficiency of the UC-protocols of [GMY04] when trimmed down to a stand-alone setting (replacing, e.g., the use of Ω-protocols by Σ-protocols).

If the parties are committed to the inputs of the OT protocol but there is no commitment to chooser's output we refer to this variant as Verifiable OT (VOT) in this paper. In this direction, Cachin and Camenisch [CC00] as well as Jarecki and Shmatikov [JS07] present protocols for VOT in 2 rounds. These protocols can be converted into COT by requesting the chooser to recommit to its received value and to prove the validity of this commitment w.r.t. the commitments for the inputs. In general, this incurs one extra communication round.

Lipmaa [Lip03] also presents a protocol under the name *verifiable homomorphic oblivious transfer* for strings. However, verifiability is defined in a different sense. The chooser will get commitments to all inputs of the sender which can later be used and referred to by the surrounding protocol. Similarly, the sender gets an encryption of the chooser's input. Hence, this is yet another form of OT,

which is related to COT and VOT, and is somewhat similar to the notion of "committing OT", introduced later in [KS06].

Recently, Camenisch *et al.* [CNs07] presented a protocol for adaptive OT, in which a sender has a list of messages and a receiver adaptively chooses one message after the other. To prevent the so-called selective-failure problem mentioned in [CNs07] (which is similar to the problem discussed in [KS06], see below), the sender is required to commit to its input list of messages and to prove consistency w.r.t. this list in all ensuing runs of adaptive OT.

More generally, we note that standard OT protocols, which are secure in a stand-alone setting, must be carefully dealt with when used as subroutines in higher level protocols. Kiraz and Schoenmakers [KS06] show that there are actually several protocols in the literature (e.g., [Pin03, MNPS04, MF06]) where the use of standard OT compromises the overall security of the protocol. Namely, a malicious sender may put 'bogus' values instead of the correct messages, and by doing so, compromise the privacy of the surrounding protocol. The use of COT or VOT protocols may prevent such problems.

Our Contributions

We present a protocol that implements COT, assuming that a (2,2)-threshold homomorphic cryptosystem has been setup before (as in, e.g., [CDN01]). This setting also allows for multiple (sequential) runs of the COT protocol, amortizing the initial cost of setting up the (2,2)-threshold cryptosystem. Our COT protocol efficiently transfers bit strings. Using the random oracle model our protocol achieves 2 rounds of interaction.

Compared to the COT protocol of [GMY04], which works for bits only, the cost of transferring $O(k)$-bit strings (for security parameter k) using our protocol is comparable to the cost of transferring a single bit using the protocol of [GMY04]. Compared to the 2-round VOT protocol of [JS07] for bit strings, which can be turned into a 3-round COT protocol (see above), our protocol uses one round less and is also computationally more efficient. However, [JS07] only assumes a common reference string (CRS) containing an RSA modulus (among other things), while we assume that a (2,2)-threshold homomorphic cryptosystem has been setup.

The security analysis of our COT protocol is done using the simulation paradigm, in the model described by [Lip03]. Although the privacy for both parties is computational (as the commitments in our protocol are public key encryptions), we show a simulation which produces a statistically indistinguishable view of the COT protocol for both parties. Hence, the COT protocol does not divulge any information beyond what can be inferred from the encryptions (which are used as computationally hiding commitments).

Organization of the Paper

The rest of the paper is as follows. In Section 2, we give some notation and definitions which are used throughout the paper. In Section 3, we present our COT protocol together with a proof of security. In Section 4, we discuss the

complexity of our protocol and compare with previous solutions. In Section 5, we discuss some applications which have some issues with the use of OT and motivate COT instead and finally we conclude the paper.

2 Preliminaries

Threshold homomorphic cryptosystems. Our results apply to any threshold homomorphic cryptosystem. Briefly, let $E(m, r)$ denote the encryption of value m using randomness r for a semantically secure public key encryption scheme. Often the randomness is omitted in the notation, writing $E(m)$. A cryptosystem is additively homomorphic if the product $E(m_1)E(m_2)$ results in $E(m_1 + m_2)$. As a consequence, for any public constant c, $E(m)^c$ is an encryption whose plaintext is cm.

In a (t, n)-threshold cryptosystem there are n parties, each of them holds a share of the overall secret key. There is a public key which allows anyone to encrypt messages. If at least t parties cooperate, any encryption can be successfully decrypted, whereas any collusion of less than t parties cannot get any information about the plaintext.

There are various instances of threshold homomorphic cryptosystems. The most widely used are (based on) ElGamal or Paillier. Threshold homomorphic ElGamal has the drawback of only allowing decryption of values belonging to a relatively small set, for which it is feasible to compute discrete logs. On the other hand, Paillier does not have this problem and allows decryption of encrypted values in an arbitrarily large set (e.g., 1024-bit integers). However, the distributed key generation protocol for threshold Paillier is very expensive compared to that for threshold ElGamal. It is also possible to use an amalgam of ElGamal and Paillier cryptosystems: the key generation protocol it is that of ElGamal, while allowing decryption of full-size plaintexts like in Paillier. One drawback of the latter is that the security of the cryptosystem relies on two computational assumptions (see [DJ03]).

Σ-protocols. A Σ-protocol for a relation $R = \{(v; w)\}$ is a 3-round protocol between a prover and a verifier, where the prover acts first. Both parties have the value v as common input, and the prover has a *witness* w as private input, where $(v; w) \in R$. A Σ-protocol is a proof of knowledge for relation R which satisfies special soundness and (special) honest-verifier zero-knowledge. See [CDS94] for further details. Moreover, non-interactive Σ-proofs are easily obtained in the random oracle model.

We will also use the fact that both for homomorphic ElGamal encryptions and for Paillier encryptions, there are efficient Σ-protocols for the relation $R_{\mathsf{enc}} = \{(e; m, r) : e = E(m, r)\}$, proving knowledge of the message m and randomness r for a given encryption $e = E(m, r)$.

(Non-Interactive) Public and Private Threshold Decryption. Given a ciphertext in the (t, n)-threshold cryptosystem, at least t parties willing to decrypt, produce

shares of the decryption, based on their respective shares of the secret key. This information is broadcast and with this, everyone can simply recover the plaintext by using a reconstruction algorithm. Putting this more formally, on ciphertext c, at least t parties broadcast $c_i = D_{sk_i}(c)$, where sk_i denotes the secret key share for the i-th party. Later, everyone can perform $m = R(c_1, \ldots, c_t)$ where $c = E(m)$, where R denotes the public reconstruction algorithm.

In order to withstand malicious adversaries, parties have to prove that the decryption share c_i is correctly computed. For this, they use a Σ-protocol for the relation $R_{\mathsf{tdec}} = \{(c_i, c; sk_i) : c_i = D_{sk_i}(c)\}$.

For the security of this process, and for later use in our security proofs, we assume that if $t-1$ parties are corrupted, then there is a simulator that on inputs $e = E(m, r)$, the message m, and the $t-1$ shares of the private key for the corrupted parties, it can produce a statistically indistinguishable view of the decryption protocol. The concrete details on how to do this depend on the specific threshold encryption scheme used. For examples, see [ST04, Section 2] for the homomorphic threshold ElGamal, and [DJ01, Section 4.1] for the threshold Paillier cryptosystem.

In our protocol, we consider a variant of the threshold decryption protocol, the so-called *private threshold decryption* [CDN01, full version]. Here, the requirement is that one of the t parties will be the only party who will recover the secret. This is easily achieved: all $t-1$ other parties follow the protocol, and broadcast their shares (along with the proofs of correctness). The party who will learn the plaintext proceeds with the decryption process privately, collects all decryption shares from the $t-1$ other parties, and privately reconstructs the message. Note that the remaining parties will not get any information about this message.

Secure multi-party computation from threshold homomorphic cryptosystems. Cramer, Damgård and Nielsen [CDN01] present a framework to build secure multiparty computation of any functionality that can be expressed as an arithmetic circuit (or formulae). Roughly, on inputs $e_1 = E(m_1)$ and $e_2 = E(m_2)$ where m_1 and m_2 may be unknown to everybody, parties can compute $E(m_1 + m_2)$ without interaction because of homomorphic properties. To compute $E(m_1 m_2)$ parties must engage in a *secure multiplication* protocol. If in the latter case one of the values, say m_1, is private to one of the parties, this party can compute $e = e_2^{m_1} E(0, r)$ proving that this is the case. Clearly, e encrypts $m_1 m_2$. This protocol is usually referred to as *private-multiplier* (see, e.g., [ST04]). For later use in the paper, the relation for the proof given in the private-multiplier gate is denoted as $R_{\mathsf{pm}} = \{(e_1, e_2, e; m_1, r_1, r) : e_1 = E(m_1, r_1) \wedge e = e_2^{m_1} E(0, r)\}$.

For later use in the simulation of our protocols, given $E(m_1)$, $E(m_2)$ and $E(m_1 m_2)$, the private-multiplier gate can be statistically simulated when there are at most $t-1$ corrupted parties in a (t, n)-threshold homomorphic cryptosystem. For details, see [CDN01, DJ01, ST04, DN03].

Encryptions as Commitments. A probabilistic public key encryption scheme can be used as a non-interactive commitment scheme. One party commits to a

message by encrypting it. The opening is done by disclosing the message and the randomness used.

In this scenario we have to be careful: the holder of the private key can *always* see the contents of any commitment of this type and, depending on the encryption scheme used, this party might recover the randomness and therefore virtually open *any* commitment.[1] This compromises the hiding property for the commiter that do not know the secret key.

We can resolve this issue with the following two possible actions: using the encryption scheme as a commitment without allowing any of the parties to know the secret key; while another suitable alternative could be to set up a threshold encryption scenario. In this way, the ability to decrypt can be distributed in a threshold fashion (possibly letting the threshold be the total number of parties).

Given a commitment $e = E(m, r)$, its committer in this scenario is the party that knows both the message m and the randomness r. Note that parties can run private threshold homomorphic decryption w.r.t. one party to retrieve the message behind e, but this not always allows the recipient to obtain the randomness used in e (e.g., ElGamal), and therefore this party will not be able to open e as a commitment. If party P is the committer of $e = E(m, r)$ we denote it by $e = \text{commit}_P(m, r)$.

Committed Oblivious Transfer. In 1-out-of-2 OT there are two parties: the sender S, who inputs two private values s_0 and s_1; and the chooser C who has a selection bit b. At the end, C receives the value s_b. The main security requirement of any protocol implementing OT is that after running the protocol S will not gain any information about the C's selection bit b; and C will be ignorant about the value of s_{1-b}.

Definition 1. *A COT protocol is run between the parties S and C. At the beginning there public commitments $\text{commit}_S(s_0, r_0)$, $\text{commit}_S(s_1, r_1)$, and $\text{commit}_C(b, r)$. S inputs s_0, s_1 and r_0, r_1, while C inputs b, r. At the end of the protocol, C receives s_b and a fresh commitment $\text{commit}_C(s_b, u)$ is publicly available. S learns nothing about b while C has no clue about s_{1-b} (See Figure 1).*

Now we point out the difference between COT and VOT in more detail. COT and VOT are identical except that in VOT the commitment by the chooser to its selected value s_b is not required. Keeping this in mind, we notice that [Cré90, CvdGT95, CD97, GMY04] are papers that present COT (of bits). Instead, in [CC00, JS07] only VOT protocols are presented. However, the different use of these terms causes some confusion: Crépeau [Cré90] introduces COT under the name of VOT, Jarecki and Shmatikov [JS07] present protocols for VOT, while they use the term COT. In the latter paper, they present a UC-secure VOT protocol (which for them is a COT), modifying the definition of the ideal functionality for COT by Garay *et al.* [GMY04] to make it into VOT. It

[1] For Paillier encryptions, one is able to recover both the plaintext and the randomness used if one knows the private key. Whereas, for ElGamal encryptions recovering the randomness is impossible under the DL assumption.

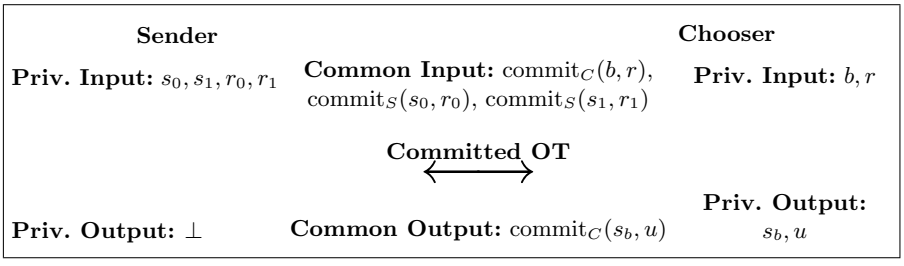


Fig. 1. Committed Oblivious Transfer

is straightforward to see that in line with our definitions COT implies VOT by just ignoring the output commitment. Vice versa, a VOT protocol can be turned into a COT protocol by adding a round in which the chooser commits to the received bit string and proves the validity of the commitment.

Security definitions. The main security obligation is to show that our protocol achieves the privacy requirements for COT. There are protocols in the literature that achieve unconditional privacy for one of the parties (e.g., [NP01, Tze02, Lip03]) while the privacy for the other party on a computational assumption. As our commitments are encryptions of the underlying threshold public key cryptosystem, we can only give computational privacy to both parties. However, our protocol achieves more than computational privacy: we show that for any corrupted party (sender or chooser) there exists a simulator that produces a view of the protocol which is statistically indistinguishable from the view of the corrupted party executing a real instance of the protocol. This has clear consequences in the framework of [CDN01]: a successful attacker to our protocol is an attacker to the security of underlying cryptosystem without loss in its success probability. This results in modular security proofs of higher level protocols that use our COT as a subroutine.

To carry out such simulations, we proceed as follows. Assuming that one party is corrupted, we build an efficient simulator that has access to the public input, private secret shares of secret key and, as done in [Lip03], the private output in the case that the chooser is corrupted. Besides, the simulator knows the public output.

3 Committed Oblivious Transfer Protocol

In this section, we will present our COT protocol. A (2,2)-threshold homomorphic cryptosystem is assumed to be set up. We let E denote the encryption algorithm of this cryptosystem, and as explained above, we also use E as a non-interactive commitment scheme.

Let $e_0 = E(s_0, r_0)$ and $e_1 = E(s_1, r_1)$ be the commitments to the sender's input strings s_0 and s_1, and $e = E(b, r)$ be the commitment to the chooser's selection bit b.

Using the general approach to secure multiparty computation of [CDN01], the COT protocol corresponds to the secure evaluation of an arithmetic circuit given by $t = b(s_1 - s_0) + s_0$ which clearly returns s_0 if $b = 0$ and s_1 when $b = 1$. This approach is so general that even s_0, s_1 and b need not be known to any party. Note that the output of the evaluation will be an encryption $e' = E(t) = E(s_b)$. If inputs (s_0, s_1) and/or b are known to the respective parties then one can securely compute e' using a private-multiplier gate (instead of a secure multiplication gate), resulting in a more efficient protocol.

Once e' is obtained, according to one of the COT requirements, only the chooser must recover the plaintext. For this, we use *private decryption*, where the chooser is the one who will learn the plaintext inside e'.

To complete the COT protocol, the chooser needs to commit to the received value s_b, and to prove that it does so correctly. In principle, this can be done using some proofs of knowledge. However, we will use the fact that our commitments are encryptions for a threshold cryptosystem: to prove that a fresh commitment e'' to output s_b is correct, we observe that this proof equivalent to show that e''/e' is an encryption of 0. The latter statement is proved by actually decrypting e''/e'.

As a final remark, we see that if the chooser starts producing e', it turns out that it has to wait for the decryption share of it from the sender, so that it later can produce the fresh commitment as just explained. This results in at least 3 rounds of communication. However, if the sender starts, it produces e' and at the same time the decryption share for e', which reduces the overall strategy to at least 2 rounds of communication. In both cases, the computational cost is actually the same. For this reason, we only go into the details of this second approach, as it results in a more efficient way of doing COT.

2-Round COT Protocol

We now present our protocol for COT. This protocol has two rounds and it is quite efficient compared to the state of the art. In the beginning of the protocol, we take advantage of the fact that the values for the commitments e_0, e_1 and e are known to the respective parties. The protocol is as follows.

Step 1. The sender produces $e' = E(b(s_1 - s_0) + s_0) = e^{(s_1 - s_0)} \cdot e_0 \cdot E(0, r')$ and the Σ-proof for relation R_{pm} on $(e, e_1/e_0, e'/e_0; s_1 - s_0, r_1 - r_0, r')$. The sender also produces its decryption share s_S of e', along with the Σ-proof for relation R_{tdec} on $(e', s_S; sk_S)$.

Step 2. After checking the two proofs given by the sender, and if they pass, the chooser then produces its corresponding decryption share for e', denoted as s_C. Combining s_S and s_C, the chooser gets s_b. Immediately, the chooser produces a fresh encryption $e'' = E(s_b, u)$ for a fresh random u, and generates its decryption share for e''/e', denoted as $\hat{s}_C$. Then, e'' and $\hat{s}_C$ are sent along with the Σ-proofs for R_{enc} and R_{tdec} on inputs $(e''; s_b, u)$ and $(e''/e', \hat{s}_C; sk_C)$ respectively.

Step 3. Finally, upon receiving e'', the sender produces its decryption share for e''/e', denoted as $\hat{s}_S$. This is combined with $\hat{s}_C$ to check whether the resulting

decrypted value is 0. If so, the sender accepts e'' as a valid commitment for the chooser's output. Otherwise, the sender rejects.

The protocol is sketched in Figure 2.

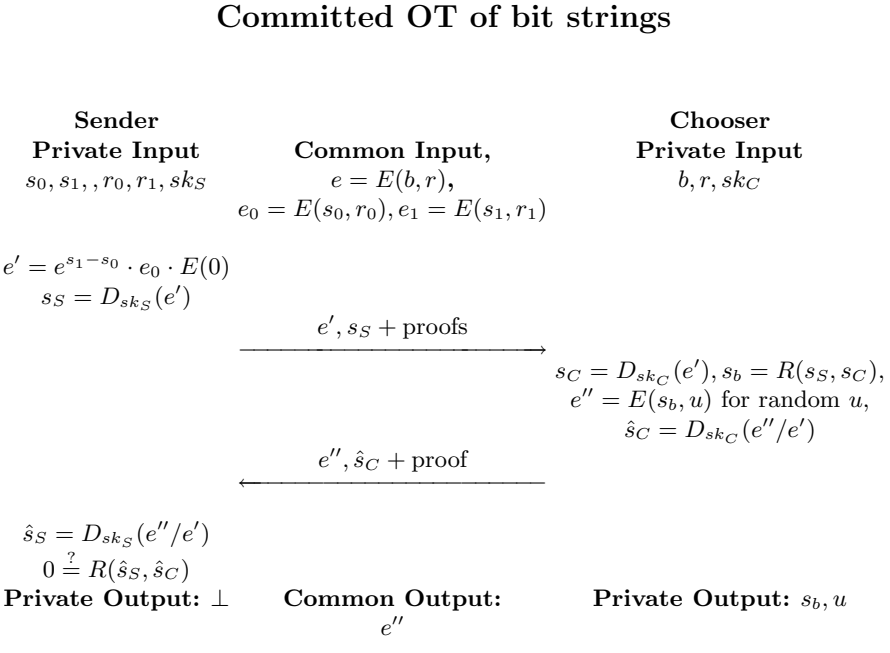


Fig. 2. Sketch of the committed OT protocol

The value s_b denotes the output of the chooser after privately decrypting e'. When this value has been computed, a fresh commitment to s_b (denoted as e'') by the chooser has to be sent in order to fulfill the COT requirement that the chooser's output must be committed. Notice here that without the fresh commitment to s_b the protocol fulfill the VOT requirement in one round only.

Security Analysis

For the security analysis, we are going to prove that this protocol fulfills the privacy requirements for COT. We are going to show that given a party is corrupted, there exists a simulator that can produce a view which is statistically indistinguishable from the view of that party interacting with the other honest party.

Before starting the proof we make some remarks in the security model to make the proof precise. As we mentioned earlier, before the simulation is run the simulator already knows the shares of the secret key of the corrupted party. The reason is that the threshold cryptosystem is set up before the protocol starts, and therefore we assume that the simulator extracts this information when the distributed key generation is run.

Also, in case the chooser is corrupted, we use the approach in [Lip03]: the simulator will be given access to the received value by the chooser. From this and public information, we construct a simulator that produces an indistinguishable view for the adversary w.r.t. the view in the real execution.

Finally, we remind that the protocol gives computational privacy to both parties, the sender and the chooser, because of the semantic security of the underlying cryptosystem. Going a bit further than computational privacy, we now show that the protocol is simulatable for both parties and those simulations produce views which are statistically indistinguishable from the views in the real protocol executions.

Theorem 1. *On the sender's inputs s_0, s_1 (and randomness r_0 and r_1), the chooser's private selection bit b (and randomness r), where public commitments to the parties' inputs $e_0 = E(s_0, r_0)$, $e_1 = E(s_1, r_1)$, and $e = E(b, r)$ are available, the COT protocol privately gives s_b (and a fresh randomness u) to the chooser, along with a public commitment $e'' = E(s_b, u)$.*

Proof. As we argued before, we assume that one of the parties is corrupted. Based on public information besides of its private decryption share, we show a simulation which produces a view to the adversary that is statistically indistinguishable from the view in the real protocol execution.

In all cases, a set of valid public inputs is available: e is a commitment to the chooser's selection bit, and e_0, e_1 are respective commitments to the sender's inputs. Also, the simulator is assumed to get the public output commitment e'' which is a valid commitment to chooser's received value.

Case 1- The chooser is corrupted. We first prove the security for the case that the chooser is corrupted. The simulator has the chooser's private key share sk_C, and received value s_b, apart from the public commitments. From this information, the simulator constructs a view for the chooser which is statistically close to the one when interacting with the honest sender.

The simulator proceeds as follows:

1. The simulator computes $e' = e'' \cdot E(0)$. The value e' together with a simulation of the private-multiplier gate (over multiplicands e and e_1/e_0 and result e'/e_0) are output.
2. At the same time, the decryption share s_S can be simulated given e', its plaintext (which is s_b) and the share of private key sk_C of the chooser. All proofs at this stage are also simulated.

This completes the simulation for the malicious chooser. The transcript is consistent and statistically indistinguishable from the chooser's view when interacting with the honest sender.

Case 2- The sender is corrupted. We next prove the security for the case that the sender is corrupted. The simulator has only sender's private key share sk_S and all public information as described above. From this information, the simulator

constructs a view for the sender which is statistically close to the one when interacting with the honest chooser.

The simulator proceeds as follows:

1. The simulator waits until the sender produces the encryption e' and the decryption share for e'. The simulator checks all the proofs as if the honest chooser would check in the real protocol execution. If all proofs are passed, the simulator goes on, otherwise it aborts.
2. Now, simulator prepares e'' as $e' \cdot E(0)$ and outputs it along with a simulated proof of knowledge. Also, it simulates $\hat{s}_C$ calling the simulator to the decryption process on inputs e''/e', plaintext 0 and the sender's secret key share sk_S.

This completes the simulation for the malicious sender. The transcript is consistent and statistically indistinguishable from the sender's view when interacting with the honest receiver. □

4 Complexity Analysis and Comparison

Our protocol involves only a constant computational, communication and round complexities. When studied in similar frameworks, our protocols are as efficient as the COT protocol by Garay *et al.* [GMY04] which is the most efficient one up to now. We stress that the protocol of Garay *et al.* works in a stronger model, since they are interested in the UC framework. We, instead, will adapt their protocol to our framework to be able to carry out a comparison.

In the following, we present the precise description for the complexity of our protocol. For a concrete result we use (2,2)-threshold ElGamal cryptosystem by considering offline computations. In the protocol by Garay *et al.* they need Ω-protocols for the proofs of knowledge. For simplicity, we trimmed them down to the simpler Σ-protocols. This is done to make a reasonable comparison.

COT protocol by Garay et al. Let's roughly sketch the protocol idea. The CRS consists of the pair (g, h), where nobody knows the discrete log x of h to the base g, i.e. $h = g^x$. The protocol uses Pedersen commitments, and so, let $E_0 = g^{r_0} h^{s_0}$ and $E_1 = g^{r_1} h^{s_1}$ denote the commitment to sender's inputs s_0 and s_1. Also, $E = g^r h^b$ is the commitment to chooser's input b. The protocol has the following two main steps:

1. The sender "re-encrypts" E_0 and E_1 under the 'keys' E and E/h respectively. Denote E'_0 and E'_1 the resulting encryptions. Note that E_b will be re-encrypted with the key g^r. It also proves that this is done correctly.
2. The chooser can only "decrypt" the message in E'_b as it knows the secret exponent r, recovering s_b. On the other hand, the chooser cannot decrypt E'_{1-b} unless the discrete-log of h to the base g is known. To finish, the chooser has to recommit to the received value s_b and prove that this is the case.

See [GMY04] for more details. In the first step, for the reencryption, 4 exponentiations are computed by the sender (2 of them can be off-line). The proofs at that step cost 16 exponentiations (8 of them can be off-line). As for the second step, the chooser needs only 1 on-line exponentiation to retrieve the chosen value. To finish, the chooser computes a fresh commitment which costs 1 off-line exponentiation. The proof of knowledge at the end costs 8 exponentiations (4 can be off-line). In total, there are 15 on-line and 15 off-line exponentiations.

VOT by Jarecki and Shmatikov. We now just sketch the VOT protocol in [JS07]. The input commitments are encryptions under a homomorphic public key cryptosystem (the public key is part of the CRS). The chooser first sends a new public key together with the encryption of its selection bit under this new cryptosystem, proving that this is done correctly. Later, the sender encrypts its inputs under this new public key, combining them with the encryption for the selection bit. Finally, the chooser can decrypt both ciphertexts, but only one of them contains the selected value, and the other one is random.

To convert it into COT, the chooser must recommit to its received value, producing a proof that the value encrypted is consistent with previous commitments. This protocol results in 3 rounds.

This scheme virtually works for any homomorphic encryption. When instantiated to additively homomorphic ElGamal, for the sake of our comparison, the protocol is slightly less efficient than that of [GMY04], around 17 on-line and 16 off-line exponentiations (mainly due to the generation of the new cryptosystem, the recommitment of the selection bit and the respective proofs of knowledge). Meanwhile, for the VOT protocol, the cost is 13 and 10 on-line and off-line exponentiations, resp.

Our COT protocol. Now we present the computational cost for our protocol in the case of (2,2)-threshold ElGamal. In Table 1 we study the computational complexity of the building blocks used in our protocol. For the private-multiplier gate, we include the cost of producing the output plus the Σ-proof for relation R_{pm}. In the case of the private threshold decryption, we include the costs for generation of the decryption shares and one Σ-proof for R_{tdec}. And finally, we consider the recommitment at the last step. Concretely, in the case of e'', chooser has to encrypt to the received value plus the Σ-proof for the knowledge of the randomness used in that encryption. This suffices as if chooser passes this proof and e'/e'' decrypts to 0, it implies it knows the plaintext in e''. We divide the complexities analysis into on-line and off-line computations.

To get the total number of exponentiations, we note that our protocol requires one private-multiplier gate at the first step (to produce e'), two private threshold decryptions (for decrypting e' and e''/e') and one encryption at the last step (to generate e''). Therefore, we have in total 12 on-line and 12 off-line exponentiations.

Observe that the way of proving that the fresh commitment is correct in our protocol is different (yet equally efficient as the proof in [GMY04]). The protocol in [GMY04] needs 9 exponentiations to recommit and prove. Our needs

Table 1. Number of exponentiations of building blocks used in our COT protocol for (2,2)-threshold ElGamal setting

	Online	**Offline**
Private-multiplier gate	3	5
Private threshold decryption	4	2
Recommitment	1	3

9 exponentiations as well: produce e'' and one threshold decryption. technique we use is a little bit different from the general zero knowledge proofs.

If we restrict ourselves to a VOT protocol, removing the recommitment step, we can see our protocol is really much more efficient than the current protocols in the state-of-the-art. It certainly requires 7 on-line and 7 off-line exponentiations (against 11 and 10 resp. for Garay *et al.*'s protocol) and also in only one round of interaction. Ours easily generalizes to any (2,2)-threshold homomorphic cryptosystem at a cost of a distributed key generation protocol at the beginning.

5 Concluding Remarks

As we mentioned before there is a generic attack that can be produced when oblivious transfer is used as a building block for higher level protocols implementations. Kiraz and Schoenmakers [KS06] present that there are several protocols for secure two-party computation using Yao's garbled circuit in the presence of malicious adversaries [Pin03, MNPS04, MF06] which have a security issue with the use of standard OT, and COT is presented as a direct solution. Generally, the problem arises due to the fact that there is no connection between the intermediate outputs in the protocol to the ones that are input for the OT protocols. We note that COT protocols (or any other combination of OT and commitments) may be therefore better to use within larger protocols assuming the correctness of the values inside the commitments. This correctness is controlled by the surrounding protocol and not by the COT protocol.

Moreover, there are a number of protocols for standard OT over bit strings which are profitable for many applications, and therefore, we stress that our COT protocol may also result in efficient implementations since it works for bit strings. Also we stress that once OT is used as a subprotocol in the semi-honest model, a COT protocol might be a good candidate to extend the higher-level protocol to the malicious case. Of course other solutions may be applicable though, but that would imply, in most of the cases, a redesign of the protocol being considered.

Finally, we highlight that our setting is quite different from the previous OT protocols. We use a $(2, 2)$-threshold setting in our protocol and of course, one might easily extend it to $(2, n)$-threshold cryptosystem. In particular it might be interesting the case $n = 3$ since still the adversary consists of only one party. The setting to adopt might clearly depend on the applications.

As further work, it could be interesting to present a protocol for committed oblivious transfer for bit strings in the universal composable framework and in the non-erasure model.

Acknowledgements. We thank Stas Jarecki and anonymous referees for valuable comments on a previous version of this paper. The work has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The third author is supported by the research program Sentinels (`http://www.sentinels.nl`). Sentinels is being financed by Technology Foundation STW, the Dutch Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

References

[Can00] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: IEEE Symposium on Foundations of Computer Science, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2000)

[CC00] Cachin, C., Camenisch, J.: Optimistic fair secure computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 93–111. Springer, Heidelberg (2000)

[CD97] Cramer, R., Damgård, I.: Linear zero-knowledge – a note on efficient zero-knowledge proofs and arguments. In: ACM Symposium on Theory of Computing – STOC 1997, pp. 436–445. ACM Press, New York (1997)

[CDN01] Cramer, R., Damgård, I., Nielsen, J.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–300. Springer, Heidelberg (2001)

[CDS94] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

[CNs07] Camenisch, J., Neven, G., shelat, a.: Simulatable adaptive oblivious transfer. In: Camenisch, J., Neven, G. (eds.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)

[Cré87] Crépeau, C.: Equivalence between two flavours of oblivious transfers. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 350–354. Springer, Heidelberg (1988)

[Cré90] Crépeau, C.: Verifiable disclosure of secrets and applications. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 181–191. Springer, Heidelberg (1990)

[CvdGT95] Crépeau, C., van de Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995)

[DJ01] Damgård, I., Jurik, M.: A generalization, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

[DJ03] Damgård, I., Jurik, M.: A length-flexible threshold cryptosystem with applications. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 350–364. Springer, Heidelberg (2003)

[DN03] Damgård, I., Nielsen, J.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)

[EGL85] Even, S., Goldreich, O., Lempel, A.: Randomized protocol for signing contracts. Communications of the ACM28, 637–647 (1985)

[GMY04] Garay, J., MacKenzie, P., Yang, K.: Efficient and universally composable committed oblivious transfer and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 297–316. Springer, Heidelberg (2004)

[JS07] Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)

[KS06] Kiraz, M., Schoenmakers, B.: A protocol issue for the malicious case of Yao's garbled circuit construction. In: 27th Symposium on Information Theory in the Benelux, pp. 283–290 (2006)

[Lip03] Lipmaa, H.: Verifiable homomorphic oblivious transfer and private equality test. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 416–433. Springer, Heidelberg (2003)

[MF06] Mohassel, P., Franklin, M.: Efficiency tradeoffs for malicious two-party computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006)

[MNPS04] Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay – a secure two-party computation system. In: USENIX Security, pp. 287–302 (2004)

[NP01] Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: 12th annual ACM-SIAM symposium on Discrete algorithms–SODA '01, pp. 448–457. ACM Press, New York (2001)

[Pin03] Pinkas, B.: Fair secure two-party computation. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. LNCS, vol. 2656, pp. 87–105. Springer, Heidelberg (2003)

[Rab81] Rabin, M.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory (1981)

[ST04] Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 119–136. Springer, Heidelberg (2004)

[Tze02] Tzeng, W.: Efficient 1-out-of-n oblivious transfer schemes. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 159–171. Springer, Heidelberg (2002)

An Efficient Certified Email Protocol*

Jun Shao[1], Min Feng[2,**], Bin Zhu[2], and Zhenfu Cao[1]

[1] Department of Computer Science and Engineering
Shanghai Jiao Tong University
200030, Shanghai, China P.R.C
[2] Microsoft Research Asia
100080, Beijing, China P.R.C
minfeng@microsoft.com

Abstract. A certified email protocol, also known as a non-repudiation protocol, allows a message to be exchanged for an acknowledgement of reception in a fair manner: a sender Alice sends a message to a receiver Bob if and only if Alice receives a receipt from Bob. In this paper, we present a novel approach to combine the authorized Diffie-Hellman key agreement protocol with a modified Schnorr signature effectively to construct our certified email protocol. Our proposed certified email protocol is an optimistic protocol, with an off-line trusted third party being involved only when a party cheats or the communication channel is interrupted during exchange. We also compare our protocol with other optimistic certified email protocols, and conclude that our certified email protocol is the most efficient optimistic certified email protocol.

Keywords: Certified Email Protocol, Fair Exchange Protocol, D Optimistic Fair Exchange Protocol.

1 Introduction

We all have the following experience: when a registered letter arrives, we receive the letter if and only if we have signed an acknowledgement of reception. The two actions, i.e., signing an acknowledgement and receiving the letter occur simultaneously. In an electronically connected world, emails are used widely. Most people prefer emails to snail mails in communicating with others due to convenience and fast delivery offered by the email. An email system should also provide the same function as the registered letter that a receiver has to sign an acknowledgement of reception before a registered email can be read. Unlike the case of registered letters, the two actions, i.e., signing and receiving, cannot occur simultaneously in an email system due to its distributed nature. The protocols used in an email system, or any protocols in general, are asynchronous by nature. How can we provide the "registered letter" service in a distributed environment? The answer is the certified email protocol, also known as the non-repudiation

* This work was done when Jun SHAO was an intern at Microsoft Research Asia.
** Corresponding author.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 145–157, 2007.

protocol[1]. A certified email protocol enables a fair exchange of a message and an undeniable receipt between two untrusted parties over a network such as the Internet.

In addition to certified emails, a certified email protocol can also be used in many other applications. One application is to secure the itinerary of a mobile agent [39], where a certified email protocol is applied between two adjacent hosts when a mobile agent passes from one host to the other. The non-deniable message and receipt offered by a certified email protocol are used to identify the origin of an attack if the itinerary of the mobile agent is altered. Another application is to encourage people to share or propagate contents such as self-created movies or advertisements with others, where a certified email protocol is used to assure that users who share contents with others would get awards by redeeming the receipts from content receivers.

Due to its usefulness, the certified email protocol has been studied widely by the cryptography research community. In fact, the problem addressed by the certified email protocol is essentially a subset of the problem addressed by the fair exchange protocol, where the exchanged items are not necessarily restricted to messages and receipts as in certified email protocols, yet other digital items can also be exchanged in the fair exchange protocol. For example, both parties can exchange signatures signed by each individual party in a fair exchange protocol. As a result, most of the techniques used in fair exchange protocols can also be used in certified email protocols.

Depending on the availability and setting of a Trusted Third Party (TTP), fair exchanges can be classified into the following four types: (1) without a TTP, (2) with an inline TTP, (3) with an online TTP, and (4) with an off-line TTP. For the first type of fair exchanges, Even and Yacobi [23] proved as early as in 1980 that it is impossible to realize fairness in a deterministic two-party fair exchange protocol. Existing protocols can provide only partial fairness: computational fairness [18,21,24] or probabilistic fairness [12,27]. These protocols are, however, too complex and inefficient to be applied in practical applications. For the second type, the TTP acts as an intermediary between the sender and the receiver, and the entire message is sent through the TTP [15,16]. An inline TTP can provide full fairness since all exchanged messages are fully controlled by the TTP. The TTP, however, may become a performance bottleneck, especially when many large messages have to forward at the same time. An online TTP is similar to an inline TTP, where the TTP must be available for the entire lifetime of the exchange. In such a setting, the TTP does not need to forward the entire message. Only the signaling information such as the cryptographic key is processed and forwarded by the TTP. For the last type, also known as the *optimistic* protocol, the TTP is involved only if one of the parties behaves maliciously or the communication channel is interrupted during execution of the exchange protocol.

[1] Some researchers [35,36] think that these two protocols are different. If we do not consider message's confidentiality, both protocols can be considered as the same since they both address the same problem: exchanging a message and a receipt in a fair manner.

This property is very desirable and practical in many applications, including the distributed environment mentioned above. The last type, therefore, has attracted more researchers' attention than the other three types. Up to now, many techniques have been proposed to address the fourth type of fair exchanges, such as the escrow and verifiable escrow scheme [5], the verifiable encryption [10,1], the verifiable confirmation of signatures [17], the convertible signatures [11,28], the designated verifier proofs [25], the cross validation [37,38], the gradational signature [32], the sequential two-party multi-signature scheme [31], the verifiable and recoverable encrypted signature [29], and the verifiably committed signatures [20].

As mentioned before, most techniques used in a fair exchange protocol can also be used in a certified email protocol. There also exist many generic certified email protocols [34,35,36], where generic encryption and signature primitives are used. These generic certified email protocols usually utilize the following approach: first encrypting a message m by a symmetric encryption scheme, then encrypting the key used in the symmetric encryption by a public key encryption scheme with the TTP's public key, and finally signing the resulting ciphertext by a signature scheme with the sender Alice's private key. When the receiver Bob receives the signature, he first checks validity of the received signature. If it is valid, he sends a receipt to Alice to indicate that Bob has received the message. Bob's interest is protected since if Alice refuses to reveal the exchanged message m, the TTP Charlie can reveal the message m for him.

The most efficient existing certified email protocol with an off-line TTP is, to the best of our knowledge, the scheme proposed in [35,36]. In this paper, we propose a novel and more efficient certified email protocol with an off-line TTP. Our scheme uses the technique of authorized key agreement. We believe that we are the first in applying this technique in a fair exchange protocol.

1.1 Our Contribution

The protocol to be proposed in the paper is a certified email protocol with an off-line TTP. The major contribution of this paper is that a novel approach is used to encrypt a message in a certified email protocol. Unlike other certified email protocols with an off-line TTP, which use the TTP's public key to encrypt a randomly selected message encryption key so that the TTP can extract the message encryption key to reveal the message in the execution of the dispute protocol, our protocol encrypts a message with a key shared between the sender and the TTP, yet without involving the TTP during the exchange. The step to apply a public key encryption scheme to encrypt the message encryption key used in other certified email protocols is removed in our protocol, resulting in a more efficient protocol.

The well-known authorized Diffie-Hellman key agreement [19] is used in our scheme to achieve the goal to share the message encryption key between the sender and the TTP. Like the protocols proposed in [34,35,36][2], our protocol is

[2] Two fair exchange protocols are proposed in [34]. One is with an online TTP, and the other with an off-line TTP. In this paper, we consider only the off-line protocol.

fair and optimistic. Compared with the existing certified email protocols [2,4, 34,32,35,36], our protocol enjoys the following properties:

Fairness: Like other certified email protocols, our protocol guarantees fairness, i.e., a malicious party cannot gain any advantage over the other party in exchange of a message and a receipt. Detailed security analysis and discussion are given in Section 3.

Optimism: Compared to the scheme proposed in [2], the TTP in our protocol is involved only when one party conducts malicious behaviors or the communication channel is interrupted during exchange. In other words, our protocol is an optimistic protocol.

TTP's Statelessness: The TTP does not need to store any state information in executing our protocol to deal with disputes between the two parties.

High Performance: Our protocol has the smallest computational and communicational cost among all certified email protocols. Comparison with typical existing certified email protocols is given in Section 4.

Note that we do not deal with the subtle issue of timely termination addressed by [5,6]. We would like to point out that the techniques used in [5,6] to deal with this subtle issue can be added easily to our protocol to resolve this problem. Furthermore, we assume that there exist reliable channels between the users and the TTP.

1.2 Organization

The rest of this paper is organized as follows. In Section 2 our novel certified email protocol is described in detail, followed by the discussion of security of our protocol in Section 3. Comparison of our protocol with the certified email protocols proposed in schemes in [35,36] is given in Section 4. We conclude the paper in Section 5.

2 Our Proposed Protocol

This paper focuses on certified email protocols without considering confidentiality of the message m. Confidentiality can be achieved easily by applying an encryption scheme to the message m if needed. Before describing our protocol, we would like to describe a modified Schnorr signature scheme [7] which will be used in our certified email protocol.

2.1 Modified Schnorr Signature Scheme

The following signature scheme is based on Schnorr signature scheme [40] which is proved to be secure against the adaptively chosen message attack in the random oracle model [14] with the discrete logarithm assumption. It consists of the following three algorithms: `Setup`, `Sign`, and `Verify`.

Setup. It takes as input a security parameter 1^k and outputs a public key $(G, q, g, H(\cdot), y)$ and a secret key x, where q is a large prime, G is a finite

cyclic group with the generator g of order q, $H(\cdot)$ is a cryptographic hash function: $\{0,1\}^* \rightarrow Z_q^*$, and $y = g^x \in G$. $\mathcal{M}$ is the domain of messages.

Sign. To sign a message $m \in \mathcal{M}$, the following operations are applied: (1) choose a random $r \in Z_q^*$, (2) compute $R = g^r \in G$, and (3) set the signature to be $\sigma = (R, s)$, where $s = r + xH(m||R||y) \bmod q$.

Verify. To verify a signature σ for message m, the verifier checks $g^s \stackrel{?}{=} Ry^{H(m||R||y)} \in G$. If the equation holds, the signature is valid and outputs $b = 1$; otherwise, the signature is invalid and outputs $b = 0$.

2.2 Our Proposed Protocol

Our certified email protocol consists of the following three sub-protocols: the `setup sub-protocol`, the `exchange sub-protocol`, and the `dispute sub-protocol`. Assume that Alice is the sender, Bob is the receiver, and Charlie is the TTP. We also assume that the public key of the Certification Authority (CA) and the three parties are known to everyone. Let m denote the sent message and let σ_B denote the receipt.

Setup Sub-protocol. In our certified email protocol, first choose the system parameters (q, G, g), where q is a large prime, and G is a gap Diffie-Hellman (GDH) group[3] with the generator g whose order is q. Then Charlie select his random private key $x_c \in Z_q^*$, and computes and publishes the corresponding public key $y_c = g^{x_c} \in G$.

Alice also selects her random private key $x_a \in Z_q^*$, and computes the corresponding public key $y_a = g^{x_a} \in G$. But, she registers her public key and her system parameter with a CA to get her certificate C_A which binds her identity ID_A with the corresponding public key (q, G, g, y_a).

Exchange Sub-protocol. In this protocol, Alice sends to Bob a message m with the message description Dsc_m[4], and receives a receipt from Bob. The

[3] We call a finite cyclic group G, with the generator g whose order is prime q, is a gap Diffie-Hellman (GDH) group if the following first problem can be solved in polynomial time but no p.p.t. algorithm can solve the following second problem with non-negligible advantage over random guess within polynomial time [30].

Decisional Diffie-Hellman Problem. Given $(g, g^a, g^b, g^c) \in G * G * G * G$, decide whether $c = ab \in Z_q^*$, where a, b, c are three random numbers in Z_q^*. If $c = ab \in Z_q^*$, then (g, g^a, g^b, g^c) is a Decisional Diffie-Hellman (DDH) tuple.

Computational Diffie-Hellman Problem. Given $(g, g^a, g^b) \in G * G * G$, compute $g^{ab} \in G$, where a, b are two random numbers in Z_q^*.

[4] The description Dsc_m will enable a human being to verify a message. A simple description is the hash value of the message. The actual description depends on the application. When used in the application to encourage sharing multimedia, Dsc_m may be a description of the multimedia content such as its title, creator, etc. We note that knowledge of the description Dsc_m does not disclose its message m.

message description Dsc_m is used to check if a decrypted message matches its description. In the following description, $(\mathcal{E}_k(\cdot), \mathcal{D}_k(\cdot))$ is a pair of symmetric encryption and decryption operations with the encryption key k. $H(\cdot)$, $H_1(\cdot)$ and $H_2(\cdot)$ are cryptographic hash functions.

1. Alice first chooses a random number $r \in_R Z_q^*$, and computes

$$R_1 = g^r \in G, \quad R_2 = y_c^r \in G, \quad R' = H(R_2), \quad k = H_1(R_2),$$

$$C = \mathcal{E}_k(m), \quad s_A = r + x_a H_2(C||Dsc_m||ID_A||ID_B||ID_C||R_1||R'||y_a) \bmod q.$$

   Alice then sends $(C_A, R_1, R', C, Dsc_m, s_A)$ to Bob, where C_A is Alice's certificate obtained with the setup sub-protocol.
2. On receiving $(C_A, R_1, R', C, Dsc_m, s_A)$ from Alice, Bob first validates Alice's certificate C_A, and then checks if the following equation holds,

$$g^{s_A} \stackrel{?}{=} R_1 y_a^{H_2(C||Dsc_m||ID_A||ID_B||ID_C||R_1||R'||y_a)} \in G.$$

   If both checks are fine, Bob sends to Alice his signature σ_B on

$$(R_1, R', C, Dsc_m, s_A, ID_A, ID_B, ID_C).$$

3. Upon receiving σ_B from Bob, Alice first validates Bob's signature σ_B. If passes, Alice sends R_2 to Bob.
4. Upon receiving R_2, Bob computes the key $k = H_1(R_2)$ and uses it to decrypt the encrypted message C previously received to obtain the wanted message $m = \mathcal{D}_{H_1(R_2)}(C)$. The decrypted message m is considered as *valid* if and only if m *does* match the message description Dsc_m previously received. If he does not receive R_2, or $R' \neq H(R_2)$, or the decrypted message m does not match its description Dsc_m, Bob can invoke the dispute protocol.

Remark 2.1. R_1 will be used as a part of the key material in the Diffie-Hellman key agreement in the `dispute protocol` to be described later, and R_2 is the resulting key of the Diffie-Hellman key agreement. (R_1, s_A) is in fact a signature on $(C, R', Dsc_m, ID_A, ID_B, ID_C)$ corresponding to the public key y_a obtained by using the modified Schnorr signature scheme. Bob's signature σ_B in Step 2 above can be any type of signature.

Alice can use receipt σ_B she receives from Bob to prove to another person John that Bob has received the message m from her with the following procedure:

- Alice sends John $(\sigma_B, R_1, R', C, Dsc_m, s_A, C_A, ID_B, ID_C, m, R_2)$
- John checks whether
  - m is consistent with Dsc_m,
  - σ_B, s_A, C_A are valid,
  - $C = \mathcal{E}_{H_1(R_2)}(m)$,
  - $R' = H(R_2)$,
  - (g, R_1, y_c, R_2) is a Decisional Diffie-Hellman (DDH) tuple.

If all the above checks pass, John is convinced that Bob indeed receives the message m from Alice.

Our protocol uses a gap Diffie-Hellman group. Alice can determine whether (g, R_1, y_c, R_2) is a DDH tuple or not by using some special algorithms such as pairing. In some applications, Alice may need to prove *only* to the TTP that Bob has received message m, i.e., John is always the TTP. In this case, the protocol is the same as described above except that the gap Diffie-Hellman group can be replaced with a finite cyclic group whose CDH problem is computationally hard[5], and we do not need to use gap Diffie-Hellman group's algorithms such as pairing to solve the DDH problem. Since the TTP already knows its own secret key x_c, TTP can determine whether (g, R_1, y_c, R_2) is a DDH tuple by checking whether $R_2 = R_1^{x_c}$ holds.

Dispute Sub-protocol. If Bob has sent his signature σ_B to Alice but has not received R_2 or the received R_2 from Alice is invalid[6], he can invoke the dispute sub-protocol and sends to Charlie $(C_A, R_1, R', C, Dsc_m, ID_A, ID_B, ID_C, s_A, \sigma_B)$. Upon receiving the data from Bob, Charlie performs the following operations:

1. Charlie first validates the received data. This step is the same as the data validation in Steps 2 and 3 in the exchange sub-protocol. Charlie aborts if the validation fails. Otherwise, he continues.
2. Charlie computes $R_2 = R_1^{x_c} \in G$, and applies the decryption operation to obtain the message $m(= \mathcal{D}_{H_1(R_2)}(C))$. If m does match its description Dsc_m and $R' = H(R_2)$, Charlie sends R_2 to Bob and σ_B to Alice.

Remark 2.2. If m does match its description Dsc_m, or $R' \neq H(R_2)$, Alice cannot use Bob's receipt to prove to others that Bob has received the message m from her since the data validation described after Remark 2.1 would fail.

3 Security Discussion

Security of our certified email protocol is analyzed with the following three lemmas:

Lemma 3.1. *The modified Schnorr signature scheme is secure against the adaptively chosen message attack with the discrete logarithm (DL) assumption in random oracle model and public key substitute attack.*

Proof. Compared with the original Schnorr signature scheme [40], the only difference in the modified Schnorr signature scheme is $H(m||R||y)$ instead of $H(m||R)$. In random oracle model [14], however, both hash oracles can choose to respond with the same output to the query to $H(m||R||y)$ on input (m, R, y) and the

[5] Note that a gap Diffie-Hellman group is always a CDH-hard group but not vice versa.
[6] A received R_2 is considered as invalid if the decrypted message m does not match its description Dsc_m, or $R' \neq H(R_2)$.

query to $H(m||R)$ on input (m, R). Since the Schnorr signature scheme is proved to be secure against the adaptively chosen message attack with the DL assumption in random oracle model [33], we conclude that the modified Schnorr signature scheme is also secure against the adaptively chosen message attack in random oracle model. According to the security analysis of [7] [Section 5], on the other hand, the modified Schnorr signature scheme can resist the public key substitute attack, i.e., there exists only a negligible possibility that a different public key can be found to satisfy the signature corresponding to a specified public key.

As a result, we conclude that the lemma holds. □

Lemma 3.2. *Assume that the Computational Diffie-Hellman (CDH) assumption holds, and the hash function $H_2(\cdot)$ is a secure one-way hash function, then only Alice and Charlie can deduce the message encryption key k which is used to encrypt the message m in our certified email protocol.*

Proof. According to Lemma 3.1, only Alice can produce a valid signature (R_1, σ). In other words, R_1 is guaranteed to be generated by Alice, i.e., no one can impersonate Alice to send a valid R_1. Since the hash function $H_2(\cdot)$ is a secure one-way hash function, the only way to deduce the message encryption key k is to deduce the value of R_2. The CDH assumption implies that it is impossible to deduce R_2 from R_1 and y_s. Therefore, no one except the person who knows r or x_s can deduce the value of k. This means that only Alice and Charlie can deduce the message encryption key k. □

Lemma 3.3. *Our certified email protocol can provide fairness.*

Proof. Based on the description presented in the above section, when the exchange sub-protocol is executed normally, i.e., when Alice and Bob are honest and the communication channel works, Bob receives the message sent by Alice, Alice receives a receipt from Bob, and Charlie is not involved. What's more, if Alice and Bob are both honest, but the communication channel does not work during the execution of the exchange sub-protocol; Alice can invoke the dispute protocol to ask for TTP's help to complete the exchange. Therefore, fairness holds in these two cases. In other cases, we are going to show that our proposed protocol can also provide fairness, i.e., Alice and Bob cannot take advantage over the other in the process of execution of our protocol even if he or she behaves maliciously. Those cases are classified into the following three cases: (1) Alice is honest, but Bob is malicious; (2) Bob is honest, but Alice is malicious, and (3) Alice and Bob are both malicious.

Case 1: *Alice is honest, but Bob is malicious.* In this case, Bob aims to obtain the message m without sending his valid receipt σ_B to Alice. In our certified email protocol, Bob may cheat in Step 2 of the `exchange sub-protocol` by not sending his valid receipt to Alice. According to our protocol, however, Alice will not send the value R_2 to Bob in this situation. Bob can obtain R_2 from Charlie by executing the `dispute sub-protocol`. But in this case, he

has to send his valid receipt to Charlie before Charlie forwards R_2 to Alice. Charlie also forwards the receipt to Alice in the `dispute sub-protocol`. Furthermore, according to Lemma 3.1, only Alice can generate a valid signature (R_1, σ). In conclusion, if Bob wants to receive m, he has to send his valid receipt to Alice, directly or indirectly. Our protocol can provide fairness in this case.

Case 2: *Bob is honest, but Alice is malicious.* In this case, Alice aims to obtain Bob's receipt σ_B without sending the message m to Bob, or to make Charlie abort in `dispute sub-protocol`. In our protocol, Alice may cheat in two steps: Step 1 and Step 3 of the exchange sub-protocol. In the former one, if Alice does not send the authorized data[7] $(C_A, R_1, R', Dsc_m, s_A)$ to Bob, Bob will not send his valid receipt to Alice. On the other hand, if Alice does not send the right[8] (R_1, R') to Bob, but Bob would send the valid receipt to Alice. Alice cannot use the received receipt from Bob to prove to others that Bob has received the right message m from her, which means the receipt Alice received is useless. Therefore Alice has to send the authorized and right $(C_A, R_1, R', Dsc_m, s_A)$ to Bob in this step. In the latter one, if Alice sends invalid R_2 to Bob or does not sends R_2 to Bob, Bob can invoke the dispute sub-protocol to get m. If the received message m does not match its description, the receipt Alice obtains from Bob is useless since she cannot prove to others that Bob has received the right message m from her. In conclusion, our protocol can provide fairness in this case too.

As a result, we finish our proof. □

4 Efficiency

In this section, we compare our proposed protocol with others. To the best of our knowledge, all the existing optimistic certified email protocols are based on public key cryptography technologies. Public key cryptography takes much longer time than symmetric key cryptography or secure hash functions. In public key cryptography, the most time-consuming operation is the modular exponentiation calculation. In fact, the ratio of the time taken for a modular exponentiation operation to the time taken for a single modular multiplication is linearly proportional to the exponent's bit length [8]. As a result, we ignore single modular multiplications and other non-public key cryptography algorithms such as symmetric encryption, symmetric decryption, and hash function in our theoretical analysis of our protocol's efficiency and comparison with other certified email protocols.

To the best of our knowledge, the two protocols proposed by Wang [35,36] are the most efficient certified email protocols previously proposed with an off-line TTP. One protocol is based on the ElGamal scheme [22] and the Schnorr scheme

[7] Authorized data means others can make sure that the data is from Alice.

[8] Right means Charlie and Alice would result in the same symmetric encryption key k, and $R' = H(R_2)$.

Table 1. Comparison of time cost of our proposed protocol with Wang's

	Wan05a	Wan05b	Ours
Step 1 of Exchange	$3EXP$	$1EXP_{RSA} + 1EV_{RSA}$	$2EXP$
Step 2 of Exchange	$2EXP + 1SGN_B$[a]	$1EV_{RSA} + 1SGN_B$	$2EXP + 1SGN_B$
Step 3 of Exchange	$1VER_B$[b]	$1VER_B$	$1VER_B$
Total of Exchange	$5EXP+$ $1SGN_B + 1VER_B$	$1EXP_{RSA} + 2EV_{RSA}+$ $1SGN_B + 1VER_B$	$4EXP+$ $1SGN_B + 1VER_B$
Prove to Other	$2EXP$	$1EV_{RSA}$	$1Pairing$[c] (or $1EXP$[d])
Dispute	$3EXP + 1VER_B$	$1EXP_{RSA} + 1EV_{RSA} + 1VER_B$	$3EXP + 1VER_B$

[a] Time taken by Bob's signature algorithm.
[b] Time taken by Bob's verfication algorithm.
[c] Time taken by a pairing computation.
[d] In this case, Alice can only prove to Charlie.

[40] (denoted as Wan05a). The other is based on RSA (denoted as Wan05b). As a result, our protocol is compared with only these two protocols in efficiency comparison. We use EXP to denote the time taken by one modular exponentiation operation that ElGamal encryption scheme or the Schnorr scheme need. EXP_{RSA} denotes the time taken by one modular exponentiation operation that RSA signature or RSA decryption needs, and EV_{RSA} denotes the time taken by one modular exponentiation operation that RSA verification or RSA encryption needs. We assume that our proposed protocol uses the same group G as the group in Wan05a, no matter it is a multiplication group of a finite field or a finite rational point group over an elliptic curve.

Table 1 shows the time cost of our proposed protocol as well as Wang's protocols. The time costs in the setup phase and the certificate verification process are ignored. From the table, our protocol saves one modular exponentiation operation in the exchange sub-protocol as compared with Wan05a. If Alice needs to prove to only the TTP that she has sent the message m to Bob, one protocol saves one modular exponentiation operation in the proving process. If Alice needs to prove to others, then our protocol needs one pairing operation, which is typically slower than the two modular exponentiation operations needed in Wan05a. Comparison of our protocol with Wan05b is more complex due to different public key cryptography systems used in the two protocols. Wan05b uses RSA. ElGamal encryption scheme and the Schnorr signature scheme used in Wan05a and ours are based discrete logarithm problem, and, as a result, can take the advantage of Elliptic Curve Cryptography (ECC) which uses much shorter keys than, and therefore much faster than RSA for the same security level. For example in [41]: RSA with 2048 bits of key length has the same security level as ECC with 224 bits of key length, and ECC-224 is 7.8 times faster than RSA-2048 in full length modular exponentiation. Therefore, our protocol is also much more efficient than Wan05b. In conclusion, our protocol is more efficient than both Wan05a and Wan05b, the two most efficient certified email protocols with off-line TTP.

5 Conclusion

In this paper, we presented a novel approach to construct a certified email protocol. This new approach is based on the authorized Diffie-Hellman key agreement. Our proposed protocol is the most efficient certified email protocol among all the existing certified email schemes in terms of the number of exponentiations and communication data. Due to its efficiency, our certified email protocol is very suitable for applications in a distributed environment.

References

1. Ateniese, G.: Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures. In: ACM CCS 1999, pp. 138–146 (1999)
2. Abadi, M., Glew, N., Horne, B., Pinkas, B.: Certificated email with a light online trusted third party: Design and implementation. In: WWW 2002, pp. 387–395 (2002)
3. Ateniese, G., de Medeiros, B., Goodrich, M.: TRICERT: Distributed certified email schemes. In: NDSS 2001 (February 2001)
4. Ateniese, G., Nita-Rotaru, C.: Stateless-recipient certified Email system based on verfiable encryption. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 182–199. Springer, Heidelberg (2002)
5. Asokan, N., Shoup, V., Waidner, M.: Optimistic Fair Exchange of Digital Signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
6. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE Journal on Selected Areas in Communication 18(4), 593–610 (2000)
7. Bao, F.: Colluding Attacks to a Payment Protocol and Two Signature Exchange Schemes. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 417–429. Springer, Heidelberg (2004)
8. Boneh, D.: Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society (AMS) 46(2), 203–213 (1999)
9. Boneh, D., Durfee, G.: Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. IEEE Trans. on Info. Theo. 46(4), 1339–1349 (2000)
10. Bao, F., Deng, R., Mao, W.: Efficient and Practical Fair Exchange Protocols. In: Proceedings of 1998 IEEE Symposium on Security and Privacy, pp. 77–85 (1998)
11. Boyd, C., Foo, E.: Off-line Fair Payment Protocols using Convertible Signatures. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 271–285. Springer, Heidelberg (1998)
12. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L.: A fair protocol for signing contracts. IEEE Transactions on Information Theory 36(1), 40–46 (1990)
13. Boneh, D., Naor, M.: Timed Commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
14. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM CCS 1993, pp. 62–73 (1993)
15. Bahreman, A., Tygar, J.D.: Certified electronic mail. In: Proceedings of the Internet Society Symposium on Network and Distributed System Security, pp. 3–19 (1994)
16. Coffey, T., Daiha, P.: Non-repudiation with mandatory proof of receipt. Computer Communication Review 26(1), 6–17 (1996)

17. Chen, L.: Efficient Fair Exchange with Verifiable Confirmation of Signatures. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 286–299. Springer, Heidelberg (1998)
18. Damgård, I.B.: Practical and provably secure release of a secret and exchange of signatures. Journal of Cryptology 8(4), 201–222 (1995)
19. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
20. Dodis, Y., Reyzin, L.: Breaking and Repairing Optimistic Fair Exchange from PODC 2003. In: ACM DRM 2003, pp. 47–54 (2003)
21. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM 28(6), 637–647 (1985)
22. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Infor. Theory 31, 469–472 (1985)
23. Even, S., Yacobi, Y.: Relations among public key signature schemes, Technical Report 175, Computer Science Department, Technion, Israel (1980)
24. Goldreich, O.: A simple protocol for signing contracts. In: McCurley, K.S., Ziegler, C.D. (eds.) Advances in Cryptology 1981 - 1997. LNCS, vol. 1440, pp. 133–136. Springer, Heidelberg (1999)
25. Garay, J., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 449–466. Springer, Heidelberg (1999)
26. Haller, N.M.: The S/KEY one-time password system. In: Proceedings of the Internet Society Symposium on Network and Distributed Systems (1994)
27. Markowitch, O., Roggeman, Y.: Probabilistic non-repudiation without trusted third party. In: Proc. of 2nd Conference on Security in Communication Networks (SCN 1999), Amalfi, Italy (1999)
28. Markowitch, O., Saeednia, S.: Optimistic fairexchange with transparent signature recovery. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 339–350. Springer, Heidelberg (2002)
29. Nenadić, A., Zhang, N., Barton, S.: A Security Protocol for Certified E-goods Delivery. In: Proc. IEEE Int. Conf. Information Technology, Coding, and Computing (ITCC'04), pp. 22–28 (2004)
30. Okamoto, T., Pointcheval, D.: The gap-problems: a new class of problems for the security of cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
31. Park, J.M., Chong, E., Siegel, H., Ray, I.: Constructing Fair-Exchange Protocols for E-Commerce Via Distributed Computation of RSA Signatures. In: PODC 2003, pp. 172–181 (2003)
32. Park, J.M., Ray, I., Chong, E.K.P., Siegel, H.J.: A Certified email Protocol Suitable for Mobile Environments. In: GLOBECOM 2003, pp. 1394–1398 (2003)
33. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
34. Imamoto, K., Sakurai, K.: A certified email system with receiver's selective usage of delivery authority. In: Menezes, A.J., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 326–338. Springer, Heidelberg (2002)
35. Wang, G.: Generic fair non-repudiation protocols with transparent off-line TTP. In: IWAP 2005, pp. 51–65 (2005)
36. Wang, G.: Generic Non-Repudiation Protocols Supporting Transparent Off-line TTP. Journal of Computer Security 14(5), 441–467 (2006)

37. Ray, I., Ray, I.: An Optimistic Fair Exchange E-commerce Protocol with Automated Dispute Resolution. In: Bauknecht, K., Madria, S.K., Pernul, G. (eds.) EC-Web 2000. LNCS, vol. 1875, pp. 84–93. Springer, Heidelberg (2000)
38. Ray, I., Ray, I., Natarajan, N.: An anonymous and failure resilient fair-exchange e-commerce protocol. Decision Support Systems 39, 267–292 (2005)
39. Borrell, J., Robles, S., Serra, J., Riera, A.: Securing the Itinerary of Mobile Agents through a Non-Repudiation Protocol. In: Security Technology, 1999. Proceedings. IEEE 33rd Annual 1999 International Carnahan Conference on, pp. 461–464 (1999)
40. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
41. Eberle, H., Gura, N., Shantz, S.C., et al.: A Public Key Cryptography processor for RSA and ECC. In: Application-Specific Systems, Architectures and Processors, pp. 98–110 (2004)

Revisiting the Security Model for Timed-Release Encryption with Pre-open Capability

Alexander W. Dent[1] and Qiang Tang[1,2]

[1] Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
[2] Departement d'Informatique, École Normale Supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France

Abstract. The concept of timed-released encryption with pre-open capability (TRE-PC) was introduced by Hwang, Yum and Lee. In a TRE-PC scheme, a message is encrypted in such a way that it can only be decrypted at a certain point in time or if the sender releases a piece of trapdoor information known as a pre-open key. This paper examines the security model for a TRE-PC scheme, demonstrates that a TRE-PC scheme can be constructed using a KEM–DEM approach, and provides an efficient example of a TRE-PC scheme.

1 Introduction

The concept of Timed-Release Encryption (TRE) is attributed to May [15]. In a TRE scheme, a message is encrypted in such a way that it can be decrypted by an authorised receiver only after a certain point in time. An unauthorised receiver should not be able to determine any information about the message from the ciphertext, and an authorised receiver should not be able to determine any information about the message before the stated release time. It is worth mentioning that some other timed primitives have been developed, for example, "price via processing" by Dwork and Naor [12], timed key escrow by Bellare and Goldwasser [1,2], and timed commitments by Boneh and Noar [5].

In the literature, there are two approaches used to construct TRE schemes. One approach is based on Merkle's time-lock puzzle technique [16] and involves encrypting the message in such a way that any computer attempting to decrypt the message will take at least a certain amount of time to solve the underlying computational problem [1,7,17]. The other approach is to use a trusted time server, which, at an appointed time, will assist in releasing a secret to help decrypt the ciphertext (e.g. [6,9,17]). Generally, time-server-based schemes require interaction between the server and the users, and should prevent possible malicious behaviour of the time server. In this paper, we shall only be concerned with public-key TRE schemes that make use of time servers.

In standard TRE schemes, the receiver can only decrypt the ciphertext at (or after) the release time. If the sender changes its mind after the ciphertext is sent, and wishes the receiver to decrypt the message immediately, then the

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 158–174, 2007.

only thing that the sender can do is to re-send the plaintext to the receiver in such a way that the receiver can immediately decrypt the message. However, in some circumstances, we may need a special kind of TRE schemes, in which a mechanism enables the receiver to decrypt the ciphertext before the release time without requiring the sender to re-send the plaintext. Recently, Hwang, Yum, and Lee [14] proposed such a scheme, which they term a *Timed-Release Encryption Scheme with Pre-Open Capability* (TRE-PC). In a TRE-PC scheme, a message is encrypted in such a way that it can only be decrypted at a certain point in time, or if the sender releases a piece of trapdoor information called a *pre-open key*. It should be infeasible for any user except for the intended receiver to determine any information about the message from the ciphertext, and the receiver should only be able to determine any information about the message after the release time or if they are given the pre-open key. In the HYL model, a trusted time server is required to periodically issue a timestamp, but real-time interaction between the trusted time server and the messege senders is not needed.

Rivest, Shamir, and Wagner gave a number of applications of Timed Released Encryption including electronic auctions, key escrow, chess moves, release of documents over time, payment schedules, press releases and etc. [17]. As a special type of TRE scheme, a TRE-PC scheme is always a possible substitute of a standard TRE scheme in all the possible applications where the latter is used. In fact, we can argue that TRE-PC scheme is more suitable than the general TRE scheme in most of these applications. Taking the electronic auction as an example, normally bidders in an auction seal their bid so that it can be opened after the bidding period is closed. However, if a bidder wishes to confirm their bid to the auctioneer at some point before the pre-defined open time, then they may come across some problems if a standard TRE scheme is adopted. Document escrow provides another useful example. Many legal systems require that classified governmental information is disclosed after a certain period of time. This can be achieved by using a TRE-PC scheme, through which the classified information can be encrypted by the public key of a special agent which is responsible for disclosing classified information. Note that no original classified information is required to be stored, and in the case that the information needs to be prematurely released, a pre-open key can be sent to the special agent which is able to decrypt the encrypted classified information.

Our contribution. This paper makes three important contributions. First, we analyse the security model for TRE-PC schemes proposed by Hwang, Yum, and Lee [14] and show that it contains several deficiencies. We propose a new security model for a TRE-PC scheme. Second, we propose a new construction paradigm for a TRE-PC scheme based on the KEM-DEM approach of Cramer and Shoup [8,18]. We show that a TRE-PC scheme can be efficiently constructed from a TRE-PC KEM and a standard DEM. Lastly, we propose an efficient new TRE-PC KEM and prove its security in the random oracle model.

2 The Security Model for a TRE-PC Scheme

2.1 Notation

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers and $\{0,1\}^*$ the set of all bit strings. If $k \in \mathbb{N}$ then $\{0,1\}^k$ is the set of bit strings of length k and 1^k is the string of k ones. If $\mathcal{A}$ is a randomised algorithm, then $y \xleftarrow{\$} \mathcal{A}(x; O)$ denotes the assignment to y of the output of $\mathcal{A}$ when run on input x with fresh random coins and with access to oracle O; we write $y \leftarrow \mathcal{A}(x; O)$ if $\mathcal{A}$ is deterministic. If S is a finite set, then $x \xleftarrow{\$} S$ denotes the random generation of an element $x \in S$ using the uniform distribution. A function $\nu : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* if for all $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $\nu(k) < k^{-c}$ for all $k > k_c$.

2.2 The HYL Security Model

In the paper that proposes the concept of timed-release encryption with pre-open capability, Hwang, Yum, and Lee [14] propose a security model against which the security of a TRE-PC scheme could be assessed. We refer to this model as the HYL model. A TRE-PC scheme proposed in the HYL model consists of six polynomial-time algorithms. A Setup algorithm is initially executed by a trusted time server. This algorithm outputs a series of system parameters and a master key for the time server. The time server uses this master key with the $\mathsf{Ext}_{\mathsf{TS}}$ algorithm to create a "timestamp" for a time t. A user generates their own encryption and decryption keys using the $\mathsf{Gen}_{\mathsf{PK}}$ algorithm. Encryption can then be performed using the Enc algorithm and a pre-open key generated using the $\mathsf{Gen}_{\mathsf{RK}}$ algorithm. These two algorithms must both take the same randomly generated secret value v as input if the pre-open key is going to help decrypt the ciphertext. Lastly, a ciphertext can be decrypted using the Dec algorithm, either using the appropriate timestamp or the pre-open key.

The HYL security model claims to consider two types of adversary: an outsider attacker (which could be "either a dishonest time server or an eavesdropper who tries to decrypt a legal receiver's ciphertext") and an inside attacker (who tries to decrypt a ciphertext before the release time without the pre-open key). Due to size constraints, we will not reproduce the HYL security models which can be found in the full version of the paper [11]. However, we suggest that the HYL model is incomplete and does not model all of the possible attacks that can be made against a TRE-PC scheme. In particular,

1. In the HYL model, the decryption process is described by one single algorithm, which works in two different modes depending on the input. We feel it is therefore more appropriate to formalise the decryption process as two independent algorithms.
2. In the HYL model, the means by which the secret value v used by the Enc and $\mathsf{Gen}_{\mathsf{RK}}$ algorithms is generated is never specified. We consider it more appropriate to remove the concept of a secret value, and have a single encryption algorithm that outputs both a ciphertext and the pre-open key for that ciphertext.

3. In the HYL model for an inside attacker, the attacker is able to obtain a timestamp for any time period except for the release time of the challenge ciphertext. In reality a receiver will only ever attempt to mount this attack before the release time of the challenge ciphertext. Hence, the HYL model is too strict. This is a problem as it is often advantageous if the timestamp for a given time period enables the receiver to decrypt all the messages that were encrypted for release at earlier times.
4. The HYL model does not give an outside attacker access to the pre-open key. However, it is realistic to assume that an outside attacker might be able to observe the pre-open key as it is being sent to the legitimate receiver.
5. The HYL model claims that an outside attacker captures the abilities of "either a dishonest time server or an eavesdropper who tries to decrypt a legal receiver's ciphertext". However, an outside attacker is not given access to the time server's master key and therefore does not model a dishonest time server.
6. A TRE-PC scheme allows the sender to release pre-open key which enable the receiver to decrypt a ciphertext before its release time. In some circumstances, the sender may wish to make the receiver decrypt a false message different from which was originally sent, by sending a false pre-open key to the receiver. This type of attack is not considered in the HYL model.

2.3 A New Security Model for TRE-PC Schemes

We propose a new formulation and security model for a TRE-PC schemes. In our formulation, a TRE-PC scheme Π is given by six probabilistic, polynomial time algorithms:

1. Setup: Run by the time server, this setup algorithm takes a security parameter 1^ℓ as input, and generates a secret master-key mk and the global parameters $param$. We assume that all subsequent algorithms takes $param$ implicitly as an input.
2. Gen: Run by a user, this user key generation algorithm takes a security parameter 1^ℓ as input, and generates a public/private key pair (pk_r, sk_r).
3. Ext: Run by the time server, this timestamp extraction algorithm takes mk and a time t as input, and generates a timestamp TS_t for the time t.
4. Enc: Run by a sender, this encryption algorithm takes a message m, a release time t, and the receiver's public key as input, and returns a ciphertext C and its pre-open key V_C. It should be noted that initially the sender should send the ciphertext C in company with the release time t to the receiver, therefore the receiver can know the release time of C. The sender stores the pre-open key V_C and publishes it when pre-opening the ciphertext C.
5. $\mathsf{Dec}_{\mathsf{RK}}$: Run by the receiver, this decryption algorithm takes a ciphertext C, the pre-open key V_C, and the receiver's private key as input, and returns either the plaintext or an error message ($\perp$). In reality, the receiver can only run this algorithm after the sender releases the pre-open key V_C.
6. $\mathsf{Dec}_{\mathsf{PK}}$: Run by the receiver, this decryption algorithm takes a ciphertext C, a timestamp TS_t which is determined by the release time accompanied with C, and the receiver's private key as input, and returns either the plaintext or an error message ($\perp$).

In the proposed model, we consider the following four kinds of adversaries:

- Outside adversaries who do not know the master key of the time server and wish to break the confidentiality of a message.
- Curious time servers who knows the master key of the time server and wish to break the confidentiality of a message.
- Legal but curious receivers who try to decrypt the ciphertext before the release time without the pre-open key.
- Legal but malicious senders who try to make the receiver recover a false message different from which was originally sent.

This gives rise to four separate security models, shown in Fig. 1. All of these models mirror the standard definition for confidentiality in public-key encryption except for the binding model, which models the capability of an attacker to produce a ciphertext for which the two decryption algorithms return different messages. Each attacker may have access to one or more of the following oracles:

1. An Ext oracle that takes a time t as input and outputs the timestamp $TS_t = \mathsf{Ext}(t, mk)$.
2. A $\mathsf{Dec_{PK}}$ oracle that takes as input a ciphertext C and a time t, and outputs $\mathsf{Dec_{PK}}(C, TS_t, sk_r)$. Note that t need not be the "correct" release time for C.
3. A $\mathsf{Dec_{RK}}$ oracle that takes as input a ciphertext C and a pre-open key V, and outputs $\mathsf{Dec_{RK}}(C, V, sk_r)$. Note that V need not be the "correct" pre-open key for C.

For each of the IND games, a probabilistic, polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is deemed to have won if it outputs a value $b' = b$. $\mathcal{A}$'s advantage is defined to be $|Pr[b' = b] - 1/2|$. We may now formally define the security models for formalising the security against the above four types of adversaries.

Definition 1 (Outsider Security). *A TRE-PC scheme Π is said to be IND-TR-CCA$_{OS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*) or the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 2 (Time Server Security). *A TRE-PC scheme Π is said to be IND-TR-CCA$_{TS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*) or the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 3 (Insider Security). *A TRE-PC scheme Π is said to be IND-TR-CPA$_{IS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the Ext oracle on any time $t \geq t^*$ has negligible advantage.*

The use of the phrase 'CPA' in the definition of insider security may be misleading: since the attacker knows the user's secret key sk_r, the attacker does not gain

IND-TR-CCA$_{\mathbf{OS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(m_0, m_1, t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, t^*, pk_r)$
6. $b' \xleftarrow{\$}$ $\mathcal{A}_2(C^*, V_{C^*}, state; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

IND-TR-CCA$_{\mathbf{TS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(m_0, m_1, t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, mk; \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, t^*, pk_r)$
6. $b' \xleftarrow{\$}$ $\mathcal{A}_2(C^*, V_{C^*}, state; \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

IND-TR-CPA$_{\mathbf{IS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(m_0, m_1, t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, sk_r; \mathsf{Ext})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Enc}(m_b, t^*, pk_r)$
6. $b' \xleftarrow{\$} \mathcal{A}_2(C^*, state; \mathsf{Ext})$

BINDING

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(C^*, t^*, V^*) \xleftarrow{\$}$ $\mathcal{A}(param, pk_r; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

Fig. 1. Security models for a TRE-PC scheme

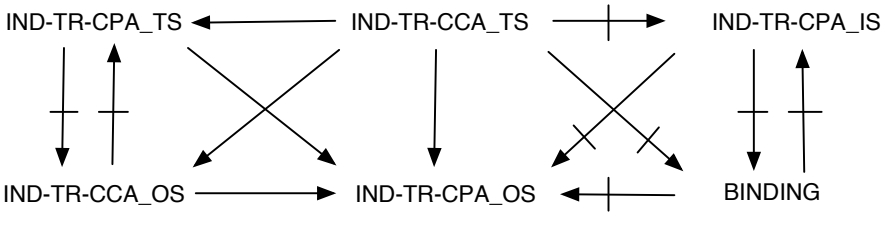


Fig. 2. Relations among the security notions

any advantage from being given access to a $\mathsf{Dec_{RK}}$ oracle or a $\mathsf{Dec_{PK}}$ oracle for any time $t < t^*$. Hence, there is no point to proposing a IND-TR-CCA$_{\mathrm{IS}}$ security model. For the other two IND security definitions, we may propose analogous CPA definitions in the usual way.

Definition 4 (Binding). *A TRE-PC scheme Π is said to be binding if, for every polynomial-time attacker $\mathcal{A}$ that outputs a triple (C^*, t^*, V^*), we have that the probability that*

$$\perp \neq \mathsf{Dec_{PK}}(C^*, TS_{t^*}, sk_r) \neq \mathsf{Dec_{RK}}(C^*, V^*, sk_r) \neq \perp$$

is negligible.

It is worth stressing that we have adopted the notation "binding" which is a property of commitment schemes such as that in [5]. The binding property for TRE-PC schemes guarantees that, if the adversary has encrypted some message then it cannot release a pre-open key to force the receiver to decrypt a false message which is different from which was original sent. It is easy to see that this is an analog to the binding property in commitment schemes. The difference is that explicit proofs are usually required in commitment schemes, while no such proofs are required in a TRE-PC scheme (as shown later in our scheme). We further point out that if the receiver obtains $\perp$ in the decryption then it can confirm that the sender has malfunctioned. The formalisation of ciphertext validity, as that in [9], is outside the scope of this paper.

In fact, the binding of a TRE-PC scheme is also concerned with the secure transportation of the pre-open key when the sender decides to open the encrypted message before the pre-defined release time. If the TRE-PC scheme is binding, then the pre-open key does not need to be integrity protected; otherwise, the pre-open key should be integrity protected to guarantee that the receiver will obtain the message which the sender has intended to send.

The relationship between these notions of security is given in Fig. 2. In this figure, "A $\longrightarrow$ B" means that if a scheme is secure in the sense of A then it is secure in the sense of B and "A$\not\longrightarrow$ B" means that we can construct a scheme which is secure in the sense of A but not secure in the sense of B. Given the relations in the figure, one can easily deduce the relation between any two security notions. Proofs of these relations can be found in the full version of the paper [11].

3 TRE-PC KEMs

The use of a symmetric encryption scheme as a subroutine of an asymmetric encryption schemes has long been known as a useful technique for improving the efficiency of asymmetric encryption. Cramer and Shoup [8,18] formalised one approach to producing such hybrid asymmetric encryption schemes. This 'KEM–DEM' approach has subsequently been applied to various other branches of asymmetric cryptography [3,4,10] and this section will explain how it can be applied to TRE-PC schemes.

A KEM-DEM scheme is composed of an asymmetric KEM and a symmetric DEM. The KEM random generates a symmetric key and an encapsulation (encryption) of that key. This symmetric key is then used by the DEM to encrypt a message. In this section, we first define a variant of KEM, namely, TRE-PC KEM, and then show that a secure TRE-PC scheme can be constructed from a secure TRE-PC KEM and a standard DEM.

3.1 Definitions of TRE-PC KEM

For the simplicity of description, the notation KEM refers to TRE-PC KEM in the following definition. A KEM consists of six probabilistic, polynomial-time algorithms:

- $\mathsf{KEM.Setup}$: This algorithm takes a security parameter 1^ℓ as input, and generates a secret master-key mk and the public parameters $param$. We assume that all subsequent algorithms take $param$ implicitly as an input
- $\mathsf{KEM.Ext}$: This algorithm takes the master private key mk and a time t as input, and generates a timestamp TS_t.
- $\mathsf{KEM.Gen}$: This algorithm takes a security parameter 1^ℓ as input, and outputs a user's public/private key pair (pk_r, sk_r).
- $\mathsf{KEM.Encap}$: This algorithm takes a release time t and a public key pk_r as input, and outputs (K, C, V_C), where K is a symmetric key, C is a ciphertext, V_C is the pre-open key of C.
- $\mathsf{KEM.Decap_{RK}}$: This algorithm takes a ciphertext C, a pre-open key V_C, and the receiver's private key sk_r as input, and returns either the encapsulated key K or an error message $\perp$.
- $\mathsf{KEM.Decap_{PK}}$: This decryption algorithm takes a ciphertext C, a timestamp TS_t which is determined by the release time accompanied with C, and the receiver's private key sk_r as input, and returns either the encapsulated key K or an error message $\perp$.

We assume that there exist a function $\mathsf{KeyLen}(\ell)$ such that the symmetric keys output by a particular TRE-PC KEM (with security parameter ℓ) are exactly $\mathsf{KeyLen}(\ell)$-bits long.

3.2 Security Definitions of TRE-PC KEM

Just as for a TRE-PC scheme, we actually define four separate security notions for a TRE-PC KEM, one for each of the different types of attacker. These security games are shown in Fig. 3. Once again, each attacker may have access to one or more of the following oracles:

1. An Ext oracle that takes a time t as input and outputs the timestamp $TS_t = \mathsf{Ext}(t, mk)$.
2. A $\mathsf{Decap_{PK}}$ oracle that takes as input an encapsulation C and a time t, and outputs either the encapsulated key K or an error message $\perp$.
3. A $\mathsf{Decap_{RK}}$ oracle that takes as input an encapsulation C and a pre-open key V, and outputs either the encapsulated key K or an error message $\perp$.

For each of the IND games, a probabilistic, polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is deemed to have won if it outputs a value $b' = b$. $\mathcal{A}$'s advantage is defined to be $|Pr[b' = b] - 1/2|$.

The formal definitions for the security of a TRE-PC KEM mirror those of a full TRE-PC scheme:

Definition 5 (Outsider Security). *A TRE-PC KEM Π is said to be IND-TR-CCA$_{OS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the* $\mathsf{Decap_{PK}}$ *oracle on the input* (C^*, t^*) *or the* $\mathsf{Decap_{RK}}$ *oracle on the input* (C^*, V_{C^*}) *has negligible advantage.*

IND-TR-CCA$_{\mathbf{OS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r; \mathsf{Ext}, \mathsf{Decap_{RK}}, \mathsf{Decap_{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(K_0, C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Encap}(t^*, pk_r)$
6. $K_1 \xleftarrow{\$} \{0,1\}^{KeyLen(\ell)}$
7. $b' \xleftarrow{\$}$ $\mathcal{A}_2(K_b, C^*, V_{C^*}, state; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

IND-TR-CCA$_{\mathbf{TS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, mk; \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(K_0, C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Encap}(t^*, pk_r)$
6. $K_1 \xleftarrow{\$} \{0,1\}^{KeyLen(\ell)}$
7. $b' \xleftarrow{\$}$ $\mathcal{A}_2(K_b, C^*, V_{C^*}, state; \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

IND-TR-CPA$_{\mathbf{IS}}$

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(t^*, state) \xleftarrow{\$}$ $\mathcal{A}_1(param, pk_r, sk_r; \mathsf{Ext})$
4. $b \xleftarrow{\$} \{0,1\}$
5. $(K_0, C^*, V_{C^*}) \xleftarrow{\$} \mathsf{Encap}(t^*, pk_r)$
6. $K_1 \xleftarrow{\$} \{0,1\}^{KeyLen(\ell)}$
7. $b' \xleftarrow{\$} \mathcal{A}_2(K_b, C^*, state; \mathsf{Ext})$

BINDING

1. $(param, mk) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$
2. $(pk_r, sk_r) \xleftarrow{\$} \mathsf{Gen}(1^\ell)$
3. $(C^*, t^*, V^*) \xleftarrow{\$}$ $\mathcal{A}(param, pk_r; \mathsf{Ext}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$

Fig. 3. Security models for a TRE-PC KEM

Definition 6 (Time Server Security). *A TRE-PC KEM Π is said to be IND-TR-CCA$_{TS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the* $\mathsf{Decap_{PK}}$ *oracle on the input (C^*, t^*) or the* $\mathsf{Decap_{RK}}$ *oracle on the input (C^*, V_{C^*}) has negligible advantage.*

Definition 7 (Insider Security). *A TRE-PC KEM Π is said to be IND-TR-CPA$_{IS}$ secure if every polynomial-time attacker $\mathcal{A}$ that does not query the* Ext *oracle on any time $t \geq t^*$ has negligible advantage.*

Definition 8 (Binding). *A TRE-PC KEM Π is said to be binding if, for every polynomial-time attacker $\mathcal{A}$ that outputs a triple (C^*, t^*, V^*), we have that the probability that*

$$\perp \neq \mathsf{Decap_{PK}}(C^*, TS_{t^*}, sk_r) \neq \mathsf{Decap_{RK}}(C^*, V^*, sk_r) \neq \perp$$

is negligible.

3.3 Construction of TRE-PC Schemes

As might be expected, we show that the combination of a secure TRE-PC KEM and a secure DEM is a secure TRE-PC scheme. We first recall the definition of a DEM [8,18]. A DEM consists of the following two polynomial-time algorithms:

- $\mathsf{DEM.Enc}$: A deterministic, polynomial-time encryption algorithm which, on the input a message m and a symmetric key K, outputs a ciphertext C.
- $\mathsf{DEM.Dec}$: A deterministic, polynomial-time decryption algorithm which, on the input a ciphertext C and a symmetric key K, outputs a message m or an error message $\perp$.

We assume that the range of possible keys K is the same as that of the associated TRE-PC KEM, i.e. $\{0,1\}^{\mathsf{KeyLen}(\ell)}$. We also assume that the TRE-PC KEM and DEM are sound in that the appropriate decapsulation/decryption algorithms 'undo' the effects of the encapsulation/encryption algorithms. We may now construct a TRE-PC scheme from a TRE-PC KEM and a DEM:

- The Setup, Ext, and Gen algorithms are given by the $\mathsf{KEM.Setup}$, $\mathsf{KEM.Ext}$, and $\mathsf{KEM.Gen}$ algorithms, respectively.
- The encryption algorithm $\mathsf{Enc}(m, t, pk_r)$ works in two steps. It first runs $(K, C_1, V_C) \xleftarrow{\$} \mathsf{KEM.Encap}(t, pk_r)$, and then computes $C_2 \leftarrow \mathsf{DEM.Enc}(m, K)$. The ciphertext is $C \leftarrow (C_1, C_2)$ and the pre-open key is V_C.
- The decryption algorithm $\mathsf{Dec}_{\mathsf{RK}}(C, V_C, sk_r)$ works in two steps. It first runs $K \leftarrow \mathsf{KEM.Decap}_{\mathsf{RK}}(C_1, V_C, sk_r)$, and then outputs the message $m \leftarrow \mathsf{DEM.Dec}(C_2, K)$. If $K = \perp$, this decryption outputs $\perp$.
- The decryption algorithm $\mathsf{Dec}_{\mathsf{PK}}(C, TS_t, sk_r)$ works in two steps. It first runs $K \leftarrow \mathsf{KEM.Decap}_{\mathsf{PK}}(C_1, TS_t, sk_r)$, and then outputs the message $m \leftarrow \mathsf{DEM.Dec}(C_2, K)$. If $K = \perp$, this decryption outputs $\perp$.

We also use the notion of one-time IND-CCA and IND-CPA security for a DEM that was proposed by Cramer and Shoup [8,18]. It is not difficult to see that we can now prove the following theorems about a TRE-PC constructed from a TRE-PC KEM and a DEM. The proofs for the IND security of the composition are similar to those of Cramer and Shoup [8,18] and can be found in the full version of the paper [11]. Note that TRE-PC KEM and and the DEM are trivially required to be sound.

Theorem 1. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is IND-TR-CCA$_{OS}$ secure and the DEM is IND-CCA secure, then the TRE-PC scheme is IND-TR-CCA$_{OS}$ secure.*

Theorem 2. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is IND-TR-CCA$_{TS}$ secure and the DEM is IND-CCA secure, then the TRE-PC scheme is IND-TR-CCA$_{TS}$ secure.*

Theorem 3. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is IND-TR-CPA$_{IS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{IS}$ secure.*

Theorem 4. *Let a hybrid TRE-PC scheme be formed from a TRE-PC KEM and a DEM. If the TRE-PC KEM is binding, then the TRE-PC scheme is binding.*

4 An Efficient TRE-PC KEM

In this section, we propose a concrete instantiation of a TRE-PC KEM. The scheme we propose shares similarities with the scheme proposed by Hwang, Yum and Lee [14]; however, our scheme is substantially simpler and, when used with a suitable DEM, gives rise to a more efficient TRE-PC scheme.

4.1 The Description

Our scheme makes use of a bilinear map on a group. In other words, we assume the existence of an instance generating algorithm that, given a security parameter 1^ℓ, outputs a group description $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are additively written groups of prime order q, P is a generator of $\mathbb{G}_1$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a polynomial-time computable, non-degenerate, bilinear map. This is normally instantiated by a super-singular elliptic curve of small embedding degree; for more details the reader is referred to the paper of Galbraith, Paterson and Smart [13].

The algorithms of the TRE-PC KEM are defined as follows:

- KEM.Setup: This algorithm takes the security parameter 1^ℓ as input, generates a group structure $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ of the required security level and chooses three hash functions:

  $$\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1 \quad \mathsf{H}_2 : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_T \to \{0,1\}^\ell \quad \mathsf{H}_3 : \mathbb{G}_1 \times \mathbb{G}_T \to \{0,1\}^{\mathsf{KeyLen}(\ell)}.$$

  The algorithm then chooses a random element $s \xleftarrow{\$} \mathbb{Z}_q$ and sets $S \leftarrow sP$. The public parameters are $param \leftarrow (\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, S)$; the master private key is $mk \leftarrow s$.
- KEM.Ext: This algorithm takes the master secret mk and a time t as input, and returns $TS_t \leftarrow s\mathsf{H}_1(t)$.
- KEM.Gen: This algorithm randomly generates $x \xleftarrow{\$} \mathbb{Z}_q$, and outputs the public/private keys $sk_r \leftarrow x$ and $pk_r \leftarrow xP$.
- KEM.Encap: This algorithm takes a release time t and the receiver's public key pk_r as input, and returns (K, C, V_C), which are computed as follows:
  1. Randomly generate $r \xleftarrow{\$} \mathbb{Z}_q$ and $v \xleftarrow{\$} \mathbb{Z}_q$.
  2. Compute $Q_t \leftarrow \mathsf{H}_1(t)$, $X_1 \leftarrow r \cdot pk_r$, $X_2 \leftarrow \hat{e}(S, Q_t)^v$.
  3. Compute $C_1 \leftarrow rP$, $C_2 \leftarrow vP$, $C_3 \leftarrow \mathsf{H}_2(C_2, X_1, X_2)$.
  4. Compute $K \leftarrow \mathsf{H}_3(X_1, X_2)$, $V_C \leftarrow vQ_t$ and $C \leftarrow (C_1, C_2, C_3)$.
- $\mathsf{KEM.Dec}_{\mathsf{RK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the pre-open key $V_C = vQ_t$, and the private key $sk_r = x$ as input, and runs as follows:
  1. Compute $X_1 \leftarrow xC_1$ and $X_2 \leftarrow \hat{e}(S, V_C)$.
  2. Check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$. If not, output $\perp$ and halt.
  3. Otherwise, return $K \leftarrow \mathsf{H}_3(X_1, X_2)$.
- $\mathsf{KEM.Dec}_{\mathsf{PK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the timestamp TS_t, and the private key $sk_r = x$ as input, and runs as follows:
  1. Compute $X_1 \leftarrow xC_1$ and $X_2 = \hat{e}(C_2, TS_t)$.
  2. Check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$. If not, output $\perp$ and halt.
  3. Otherwise, return $K \leftarrow \mathsf{H}_3(X_1, X_2)$.

4.2 Security Results

The security of our scheme is based on two principles: that it is infeasible for any attacker who does not know the private key $sk_r = x$ to compute the value $X_1 = xC_1^*$ and that it is infeasible for any attacker who does not know either the master private key, the pre-open key or the appropriate timestamp to compute the value $X_2 = \hat{e}(P, P)^{rvs}$ for the given value of C_2^*. We prove the security of our scheme in the random oracle model under the following assumptions:

Definition 9 (Computational Diffie-Hellman). *Given a group description* $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ *generated at a security level* 1^ℓ *and a pair of group elements* $(\alpha P, \beta P)$*, where* $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$*, the computational Diffie-Hellman (CDH) problem is to determine* $\alpha\beta P$*. The CDH assumption is that no probabilistic, polynomial-time algorithm can solve this problem with non-negligible probability.*

Definition 10 (Bilinear Diffie-Hellman). *Given a group description* $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ *generated at a security level* 1^ℓ *and a triple of group elements* $(\alpha P, \beta P, \gamma P)$*, where* $\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_q$*, the Bilinear Diffie-Hellman (BDH) problem is to determine* $\hat{e}(P, P)^{\alpha\beta\gamma}$*. The BDH assumption is that no probabilistic, polynomial-time algorithm can solve this problem with non-negligible probability.*

These computational assumptions allow us to prove the follow theorems about the IND security of our scheme.

Theorem 5. *The TRE-PC KEM is IND-TR-CCA*$_{TS}$ *secure in the random oracle model under the CDH assumption.*

Proof. We construct an algorithm $\mathcal{B}$ which solves the CDH problem with non-negligible probability whenever $\mathcal{A}$ breaks the IND-TR-CCA$_{\mathrm{TS}}$ security of the TRE-PC KEM with non-negligible advantage. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an IND-TR-CCA$_{\mathrm{TS}}$ attacker with non-negligible advantage. $\mathcal{B}$ runs as follows:

1. Receive an instance of the group on which the CDH problem is to be solved $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ and a CDH challenge $(\alpha P, \beta P)$.
2. Game setup: Randomly select $s \xleftarrow{\$} \mathbb{Z}_q$ and set $S = sP$. The public parameters are $param \leftarrow (\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, S)$ and the master private key is $mk \leftarrow s$. Set the user's public key to be $pk_r \leftarrow \alpha P$.
3. Phase 1: $\mathcal{B}$ executes $\mathcal{A}_1$ on the input $(param, pk_r, mk)$. $\mathcal{A}_1$ has access to several oracles during its execution (we assume, without loss of generality, that $\mathcal{A}$ never queries the random oracles with the same value twice):
   - If $\mathcal{A}$ queries the random oracle $\mathsf{H_i}$ with a new input Z, then $\mathcal{B}$ random generates a value Y from the appropriate range, stores (Z, Y) in $\mathsf{H_i}$-list and returns Y.
   - If $\mathcal{A}$ queries the $\mathsf{KEM.Decap_{RK}}$ oracle on the input $C = (C_1, C_2, C_3)$ and V_C, then $\mathcal{B}$ runs as follows:
     (a) Check whether there exists an input $Z = (z_1, z_2, z_3)$ on the $\mathsf{H_2}$-list such that $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$ and $z_3 = \hat{e}(S, V_C)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

(b) Check whether $C_3 = \mathsf{H}_2(z_1, z_2, z_3)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
(c) If both checks succeed, then $\mathcal{B}$ returns $K \leftarrow \mathsf{H}_3(z_2, z_3)$ to $\mathcal{A}$.

– If $\mathcal{A}$ queries the $\mathsf{KEM.Decap}_{\mathsf{PK}}$ oracle on the input $C = (C_1, C_2, C_3)$ and t, then $\mathcal{B}$ runs as follows:
(a) Compute $TS_t = s\mathsf{H}_1(t)$.
(b) Check whether there exists an input $Z = (z_1, z_2, z_3)$ on the H_2-list such that $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$ and $z_3 = \hat{e}(C_2, TS_t)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
(c) Check whether $C_3 = \mathsf{H}_2(z_1, z_2, z_3)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
(d) If both checks succeed, then $\mathcal{B}$ returns $K \leftarrow \mathsf{H}_3(z_2, z_3)$ to $\mathcal{A}$.

$\mathcal{A}_1$ terminates by outputting a challenge release time t^* and some state information *state*.

4. Challenge: $\mathcal{B}$ sets $C_1^* \leftarrow \beta P$ (the CDH challenge element). $\mathcal{B}$ also randomly selects $v \xleftarrow{\$} \mathbb{Z}_q$, $C_3^* \xleftarrow{\$} \{0,1\}^\ell$, $K \xleftarrow{\$} \{0,1\}^{\mathsf{KeyLen}(\ell)}$ and sets $C_2^* \leftarrow vP$, $V_{C^*} \leftarrow v\mathsf{H}_1(t^*)$. . The challenge ciphertext is set as $C^* \leftarrow (C_1^*, C_2^*, C_3^*)$.
5. Phase 2: $\mathcal{B}$ executes $\mathcal{A}_2$ on the input $(K, C^*, V_{C^*}, state)$. During its execution, $\mathcal{A}_2$ may query several oracles, these oracle queries are answered in the same way as in Phase 1. $\mathcal{A}_2$ terminates by outputting a bit b'.
6. $\mathcal{B}$ random selects an input Z on either the H_2-list or the H_3-list in such a way that all inputs are equally likely to be chosen. If $Z = (z_1, z_2, z_3)$ is an input on the H_2-list, then $\mathcal{B}$ outputs z_2. If $Z = (z_1, z_2)$ is an input on the H_3-list, then $\mathcal{B}$ outputs z_1.

We analyse this algorithm and show two things. First, that the environment that $\mathcal{A}$ can only distinguish the simulated environment from a real attack environment with negligible probability (up until the point in which $\mathcal{A}$ makes a critical query to a hash function). Second, if $\mathcal{A}$ succeeds in breaking the security of the TRE-PC KEM, then it must make a critical query with non-negligible probability and that such a critical query allows $\mathcal{B}$ to recover the solution to the CDH problem with non-negligible probability.

Suppose $\mathcal{A}$ makes at most q_i queries to the random oracles H_i, q_{RK} queries to the $\mathsf{KEM.Decap}_{RK}$ oracle and q_{PK} queries to the $\mathsf{KEM.Decap}_{PK}$ oracle. We define a critical query to be either:

– a query (z_1, z_2, z_3) to the H_2 oracle such that $z_1 = C_2^*$, $z_2 = \alpha\beta P$ and $z_3 = \hat{e}(S, V_{C^*})$, or
– a query to the (z_1, z_2) to the H_3 oracle such that $z_1 = \alpha\beta P$ and $z_2 = \hat{e}(C_2^*, TS_{t^*})$.

Note that the simulation of the random oracles is perfect up until the point where a critical query is made. Again, up until the point where a critical query is made, the simulation of the $\mathsf{KEM.Decap}_{RK}$ algorithm is perfect unless $\mathcal{A}$ submits a query (C_1, C_2, C_3) and V_C to the decapsulation oracle such that $\mathcal{A}$ has not queried the H_2 oracle on the input $z_1 = C_2$, $z_2 = r\alpha P$ and $z_2 = \hat{e}(S, V_C)$, where $C_1 = rP$, and yet $\mathsf{H}_2(z_1, z_2, z_3) = C_3$. It is clear that, since H_2 is a random oracle, these conditions will hold with probability $1/2^\ell$, which is negligible. This

argument can also be used to show that the simulation of $\mathsf{KEM.Decap}_{PK}$ is sufficient correct up until the point where a critical query is made.

Let n_2 and n_3 be the maximum number of possible entries on the H_2- and H_3-lists respectively. Note that

$$n_2 = q_2 \qquad \text{and} \qquad n_3 = q_3 + q_{RK} + q_{PK}$$

due to the way in which the decapsulation oracles are simulated. If a critical query is made, then $\mathcal{B}$ will output the solution to the CDH problem with probability at least $1/n_2 + n_3$. Let E be the event that a critical query is made and let E' be the event that the critical H_3 query is made, and note that $Pr[E] \geq Pr[E']$. Since H_3 is a random oracle, a standard argument shows that $Pr[E']$ is greater than or equal to $\mathcal{A}$'s advantage. Therefore, since $\mathcal{A}$ has non-negligible advantage, we must have that $\mathcal{B}$ has a non-negligible probability of output the solution to the CDH problem. □

Theorem 6. *The TRE-PC KEM scheme is IND-TR-CPA$_{IS}$ secure in the random oracle model under the BDH assumption.*

Proof. The proof of this theorem is similar to the proof of IND-TR-CCA$_{\mathrm{TS}}$ security, although slightly more complex. Again, we construct an algorithm $\mathcal{B}$ that solves the BDH problem with non-negligible probability whenever $\mathcal{A}$ breaks the IND-TR-CPA$_{\mathrm{IS}}$ security of the TRE-PC KEM with non-negligible advantage. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an IND-TR-CPA$_{\mathrm{IS}}$ attacker with non-negligible advantage.

Suppose that $\mathcal{A}$ makes at most q_i queries to the random oracles $\mathsf{H_i}$, and q_E queries to the $\mathsf{KEM.Ext}$ oracle. $\mathcal{B}$ will keep track of the queries made to the $\mathsf{H_i}$ oracle via a number of lists. In any execution of $\mathcal{B}$, we shall see that the $\mathsf{H_i}$-list has at most n_i elements on it, where

$$n_1 = q_1 + q_E + 1\,, \qquad n_2 = q_2 \qquad \text{and} \qquad n_3 = q_3\,.$$

$\mathcal{B}$ runs as follows:

1. Receive an instance of the group on which the BDH problem is to be solved $(\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e})$ and a BDH challenge $(\alpha P, \beta P, \gamma P)$.
2. $\mathcal{B}$ randomly choose an integer $j \xleftarrow{\$} \{1, 2, \ldots, n_1\}$. This will define $\mathcal{B}$'s guess for the challenge release time.
3. Game setup: Set $S \leftarrow \alpha P$ and define the public parameters to be $param \leftarrow (\mathbb{G}_1, \mathbb{G}_T, P, q, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, S)$. Randomly select $x \xleftarrow{\$} \mathbb{Z}_q$, set $sk_r \leftarrow x$ and $pk_r \leftarrow xP$.
4. Phase 1: $\mathcal{B}$ executes $\mathcal{A}_1$ on the input $(param, pk_r, sk_r)$. $\mathcal{A}_1$ has access to several oracles during its execution (we assume, without loss of generality, that $\mathcal{A}$ never queries the random oracles with the same value twice):
   - If $\mathcal{A}$ (or the $\mathsf{KEM.Ext}$ oracle) queries the random oracle H_1 with a new input t and this is not the j-th new query to the H_1 oracle, then $\mathcal{B}$ random generates a value $y \xleftarrow{\$} \mathbb{Z}_q$, sets $Y \leftarrow yP$, stores (t, y, Y) in H_1-list and returns Y.

- If $\mathcal{A}$ (or the $\mathsf{KEM.Ext}$ oracle) queries the random oracle H_1 with a new input t and this is the j-th new query to the H_1 oracle, then $\mathcal{B}$ adds $(t, \perp, \gamma P)$ to the H_1-list and returns γP to $\mathcal{A}$.
- If $\mathcal{A}$ queries the random oracle H_2 or H_3 with a new input Z, then $\mathcal{B}$ random generates a value Y from the appropriate range, stores (Z, Y) in the appropriate H_i-list and returns Y.
- If $\mathcal{A}$ queries the $\mathsf{KEM.Ext}$ oracle on the time t, then compute $\mathsf{H}_1(t)$, extract the appropriate element y from the H_1-list entry (t, y, Y), and returns yS to $\mathcal{A}$. If there exists no element y, i.e. if t was the j-th query to the H_1 oracle, then $\mathcal{B}$ terminates its entire execution by outputting a random group element from $\mathbb{G}_T$.

$\mathcal{A}_1$ terminates by outputting a challenge release time t^* and some state information *state*.

5. If the H_1 oracle has not been queried on t^*, then $\mathcal{B}$ "queries" H_1 on t^*.
6. If t^* is not the j-th query to the H_1 oracle, then $\mathcal{B}$ terminates its entire execution by outputting a random group element from $\mathbb{G}_T$.
7. Challenge: $\mathcal{B}$ randomly chooses $r^* \xleftarrow{\$} \mathbb{Z}_q$ and sets $C_1^* \leftarrow r^* P$. $\mathcal{B}$ sets $C_2^* \leftarrow \beta P$ and randomly selects $C_3^* \xleftarrow{\$} \{0,1\}^\ell$ and $K \xleftarrow{\$} \{0,1\}^{\mathsf{KeyLen}(\ell)}$. The challenge ciphertext is defined to be $C^* = (C_1^*, C_2^*, C_3^*)$.
8. Phase 2: $\mathcal{B}$ executes $\mathcal{A}_2$ on the input $(K, C^*, state)$. During its execution, $\mathcal{A}_2$ may query several oracles, these oracle queries are answered in the same way as in Phase 1. $\mathcal{A}_2$ terminates by outputting a bit b'.
9. $\mathcal{B}$ random selects an input Z on either the H_2-list or the H_3-list in such a way that all inputs are equally likely to be chosen. If $Z = (z_1, z_2, z_3)$ is an input on the H_2-list, then $\mathcal{B}$ outputs z_3. If $Z = (z_1, z_2)$ is an input on the H_3-list, then $\mathcal{B}$ outputs z_2.

Again we show that the environment provided by $\mathcal{B}$ to $\mathcal{A}$ almost exactly simulates the attack environment in which $\mathcal{A}$ expects to run up until either the simulation terminates because t^* is not the j-th query to H_1 or a critical oracle query is made. We define a critical oracle query to be either:

- a query (z_1, z_2, z_3) to the H_2 oracle such that $z_1 = C_2^*$, $z_2 = r^* xP$ and $z_3 = \hat{e}(C_2^*, TS_{t^*}) = \hat{e}(P, P)^{\alpha\beta\gamma}$, or
- a query to the (z_1, z_2) to the H_3 oracle such that $z_1 = r^* xP$ and $z_2 = \hat{e}(C_2^*, TS_{t^*}) = \hat{e}(P, P)^{\alpha\beta\gamma}$.

We note that the simulation of the random oracles and the extraction oracle is perfect up until the point in which a critical oracle query is made. Furthermore, we note that the probability that we make an incorrect choice of j is $1/n_1$.

Let E be the event that a critical oracle query is made and let E' be the event that the critical H_3 oracle query is made. Note that $Pr[E] \geq Pr[E']$. Now, by a standard argument, since H_3 is a random oracle, $\mathcal{A}$'s advantage in breaking the IND-TR-CPA$_{\mathrm{IS}}$ security of the TRE-PC KEM is less than or equal to $Pr[E']$. Furthermore, if E occurs, then $\mathcal{B}$ has at least a $1/n_2 + n_3$ chance of outputting the

correct solution to the CDH problem. This means that if $\mathcal{A}$ has a non-negligible advantage ϵ of breaking the IND-TR-CPA$_{\mathrm{IS}}$ security of the TRE-PC KEM, then $\mathcal{B}$ has a non-negligible probability of at least $\epsilon/n_1(n_2+n_3)$ of solving the CDH problem. □

The binding of the scheme can be proven directly in the random oracle model, or in the standard model under the following assumption.

Definition 11 (Collision Resistance). *A hash function* H *generated at security level* 1^ℓ *is collision resistance if the probability that any polynomial-time algorithm can find a pair of inputs* $x \neq y$ *such that* $\mathsf{H}(x) = \mathsf{H}(y)$ *is negligible as a function of the security parameter.*

Theorem 7. *If* H_2 *is collision-resistant, then the TRE-PC KEM is binding.*

Proof. Without loss of generality, suppose that at the end of the legitimate binding attack game the attacker outputs (C^*, t^*, V_{C^*}), where $C^* = (C_1^*, C_2^*, C_3^*)$. Recalling the definitions of $\mathsf{KEM.Decap}_{\mathsf{RK}}$ and $\mathsf{KEM.Decap}_{\mathsf{PK}}$ from the previous section, the attacker wins the game only if $O_1 \neq O_2$, where

$$O_1 = \mathsf{H}_3(X_1, X_2'),\ O_2 = \mathsf{H}_3(X_1, X_2''),\ X_1 = sk_r C_1^*,\ X_2' = \hat{e}(S, V_{C^*}),$$

$$X_2'' = \hat{e}(C_2^*, TS_{t^*}),\ C_3^* = \mathsf{H}_2(C_2^*, X_1, X_2'),\ \text{and}\ C_3^* = \mathsf{H}_2(C_2^*, X_1, X_2'').$$

If the attacker wins, then it is staightforward to verify that $X_2' \neq X_2''$; otherwise $O_1 = O_2$. Hence, if the attacker wins the game then this implies that the attacker can find a collision for H_2, where the two inputs are (C_2^*, X_1, X_2') and (C_2^*, X_1, X_2''). Under the assumption that H_2 is collision-resistant, it follows that the attacker can only win the game with a negligible probability. □

5 Conclusion

In this paper we have analysed the security model for TRE-PC schemes proposed by Hwang, Yum, and Lee, and shown its defects. We proposed a new security model which avoids the defects possessed by the HYL model. We also worked out the complete relations among the security notions defined in the proposed security model, introduced a new notion, i.e. TRE-PC KEM, and presented a hybrid model to construct TRE-PC schemes.

Acknowledgements. The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. Bellare, M., Goldwasser, S.: Encapsulated key-escrow. Technical Report Tech. Report MIT/LCS/TR-688, MIT LCS (1996)
2. Bellare, M., Goldwasser, S.: Verifiable partial key escrow. In: Proceedings of the 4th ACM conference on Computer and communications security, pp. 78–91. ACM Press, New York (1997)
3. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive: Report 2005/058 (2005)
4. Bjørstad, T.E., Dent, A.W.: Building better signcryption schemes with tag-kems. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 491–507. Springer, Heidelberg (2006)
5. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
6. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and non-interactive timed-release encryption. In: Qing, S., Mao, W., Lopez, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
7. Chan, A.C.-F., Blake, I.F.: Scalable, server-passive, user-anonymous timed release cryptography. In: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), pp. 504–513. IEEE Computer Society Press, Los Alamitos (2005)
8. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2004)
9. Crescenzo, G.D., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
10. Dent, A.W.: Hybrid signcryption schemes with outsider security. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 203–217. Springer, Heidelberg (2005)
11. Dent, A.W., Tang, Q.: Revisiting the security model for timed-release public-key encryption with pre-open capability (2006), http://eprint.iacr.org/2006/306
12. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)
13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers (2006), http://eprint.iacr.org/2006/165
14. Hwang, Y., Yum, D., Lee, P.: Timed-release encryption with pre-open capability and its application to certified e-mail system. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 344–358. Springer, Heidelberg (2005)
15. May, T.C.: Time-release crypto (1993)
16. Merkle, R.C.: Secure communications over insecure channels. Commun. ACM 21(4), 294–299 (1978)
17. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical Report Tech. Report MIT/LCS/TR-684, MIT LCS (1996)
18. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)

On the Soundness of Restricted Universal Designated Verifier Signatures and Dedicated Signatures

How to Prove the Possession of an ElGamal/DSA Signature

Fabien Laguillaumie[1] and Damien Vergnaud[2,⋆]

[1] GREYC – Université de Caen
Boulevard du Maréchal Juin - BP 5186 - 14032 Caen Cedex, France
fabien.laguillaumie@info.unicaen.fr
[2] École normale supérieure
Département d'informatique, 45 rue d'Ulm, 75230 Paris cedex 05, France
damien.vergnaud@di.ens.fr

Abstract. In 2006, Huang, Susilo, Mu and Zhang proposed the concept of *restricted universal designated verifier signatures* while Klonowski, Kubiak, Kutylowski and Lauks proposed independently the *dual* primitive of *dedicated signatures*. In both notions, a signature holder can convince one or more verifiers of his knowledge of a digital signature, but cannot exploit this knowledge without being *punished* for that. In this paper, we state that a signature holder may generically provide a proof that it has a certain signature without being punished and that consequently both primitives cannot fulfill their alleged security goals. To demonstrate the feasibility of this claim, we propose the first non-interactive universal designated verifier proof of the possession of an Elgamal or a DSA signature in the random oracle model. This construction may be of independent interest.

Keywords: Universal designated verifier signatures, zero-knowledge proof, Elgamal signatures.

1 Introduction

In order to control the dissemination of digital signatures, many schemes have been proposed to protect the privacy of the signer or the holder of a signature. We can cite for instance undeniable signatures [4], confirmer signatures [3] and designated verifier signatures [10]. In particular, some constructions aim at modifying a *traditional* signature into a signature with special properties (for instance a signature with controlled verification, like *universal* designated verifier signatures [19]).

⋆ This work was done while this author was a postdoctoral fellow in the Computer Security group of the Bonn/Aachen International Center for Information Technology.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 175–188, 2007.

In 2006, Huang, Susilo, Mu and Zhang proposed the concept of *restricted universal designated verifier signatures* while Klonowski, Kubiak, Kutylowski and Lauks proposed independently, the *dual* primitive of *dedicated signatures*. In both notions, a signature holder can convince one or more verifiers of his knowledge of a digital signature, but cannot exploit this knowledge without being *punished* for that. The main purpose of the present paper is to state that that both primitives do not achieve their purported security goals.

Background. In the electronic world, digital signatures are used to verify whether one message really comes from the alleged signer. Like handwritten signatures, standard digital signatures are *non-repudiable* and *universally verifiable*. However, universal verifiability might not suit the circumstances under which verifying signature is a valuable action.

In 1989, Chaum and van Antwerpen [4] introduced the concept of *undeniable signature* scheme in which anyone has to interact with the signer to verify the validity/invalidity of a signature. The important property of non-repudiation still holds because the signer cannot disavow a signature through a denial protocol unless the signature is indeed invalid.

Jakobsson, Sako and Impagliazzo [10] proposed *designated verifier signatures* in 1996. A designated verifier signature scheme provides authentication of a message without providing the non-repudiation property of traditional signatures and can be used to convince only one third party (*i. e.* the designated verifier and only him can be convinced about its validity or invalidity). This is due to the fact that the designated verifier can always create a signature intended for himself that is indistinguishable from an original signature.

Steinfeld, Bull, Wang and Pieprzyk [19] proposed the extended concept of *universal designated-verifier signatures* in 2003. These signatures are ordinary digital signatures with the additional functionality that any holder of a signature is able to convert it into a designated verifier signature specified to a designated verifier of his choice. Steinfeld *et al.* also showed how to construct efficient deterministic universal designated-verifier signature schemes from bilinear group-pairs and many constructions have been proposed compatible with popular digital signature schemes (*e. g.* [12,13,20,21]).

In 2006, Huang, Susilo, Mu and Zhang [8] proposed the concept of *restricted universal designated verifier signature*. In this construct, a signature holder can convince up to t verifiers that he actually knows a signature, and the convincing statement is designated to these verifiers. However, when the signature holder uses the signature $t+1$ times, then the signature becomes publicly available. This primitive can potentially be used in a service such as an Internet trial-browsing service in which a user is allowed to access the service up to t times without any charge, but will be charged on the $t + 1$ count of access.

In [11], Klonowski, Kubiak, Kutylowski and Lauks introduced a new kind of signatures, in a sense dual of the previous ones, that they called *dedicated signatures*. In their scheme, the signer can construct a signature in such a way that the recipient cannot show this signature to third parties without being punished for that. Namely, in this protocol, the signer gives the recipient a

dedicated signature; the verifier derives a standard signature from it and if the verifier presents this signature to third parties, the signature together with the dedicated signature reveal the private key of the verifier. In the present paper, we prove that these two primitives are not sound since generically they cannot fulfill their alleged security goals.

Contributions of the paper. In this paper, we prove that a signature holder may generically provide a proof that it has a certain signature without being punished and that consequently both primitives are not sound. The result comes from the well-know fact that if non uniform one-way function exist then there exists a computational zero-knowledge proof system of membership for all languages having an interactive proof system of membership [6].

As a practical example to defeat the first realization of the concept of restricted universal designated verifier signatures, we remind an efficient proof of possession of a Boneh-Lynn-Shacham signature [1] (BLS signature for short) due to Hufschmitt, Lefranc and Sibert [9].

Finally, to demonstrate the feasibility of this attack on the primary dedicated signature scheme proposed by Klonowski *et al.*, we also propose the first efficient non-interactive designated verifier proof of the possession of an Elgamal signature [5] in the so called random oracle model. This result is of independant interest, as it does not exist, as far as we know, any concrete proof of knowledge of an Elgamal or a DSA signature, but a specialization of the inefficient proofs given in [15] by Nguyen, Bao, Mu and Varadharajan.

2 Definitions

In this section, we give the definition of (restricted) universal designated verifier signature scheme and dedicated signature scheme.

2.1 Notations

The set of n-bit strings is denoted by $\{0,1\}^n$ and the set of all finite binary strings (or messages) is denoted by $\{0,1\}^*$. Concatenation of two strings x and y is denoted by $x\|y$. Let $\mathcal{A}$ be a probabilistic Turing machine running in polynomial time (a PPTM, for short), and let x be an input for $\mathcal{A}$. The probability space that assigns to a string σ the probability that $\mathcal{A}$, on input x, outputs σ is denoted by $\mathcal{A}(x)$. The support of $\mathcal{A}(x)$ is denoted by $\mathcal{A}[x]$. Given a probability space S, a PPTM that samples a random element according to S is denoted by $x \xleftarrow{R} S$. For a finite set X, $x \xleftarrow{R} X$ denotes a PPTM that samples a random element uniformly at random from X.

2.2 Universal Designated Verifier Signatures

In this subsection, we recall the definition of universal designated verifier signature (UDVS) schemes [19], together with the security model, that we will need to present the new Elgamal scheme in Section 3.

Definition 1. *A* universal designated verifier signature scheme Σ *is an 8-tuple*

$$\Sigma = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Designate}, \mathsf{Fake}, \mathsf{DVerify})$$

such that

- $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is a signature scheme:*
  - $\Sigma.\mathsf{Setup}$ *is a probabilistic polynomial-time Turing machine (PPTM) which takes an integer* k *as input. The output are the* public parameters Υ. k *is called the* security parameter.
  - $\Sigma.\mathsf{SKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{sks}, \mathsf{pks})$ *where* sks *is called a* signing secret key *and* pks *a* signing public key.
  - $\Sigma.\mathsf{Sign}$ *is a PPTM which takes the public parameters, a message, and a signing secret key as inputs and outputs a bit string.*
  - $\Sigma.\mathsf{Verify}$ *is a PPTM which takes the public parameters, a message* m*, a bit string* σ *and a signing public key* pks*. It outputs a bit. If the bit output is 1 then the bit string* σ *is said to be a* signature *on* m *for* pks.
- $\Sigma.\mathsf{VKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{skv}, \mathsf{pkv})$ *where* skv *is called a* verifying secret key *and* pkv *a* verifying public key.
- $\Sigma.\mathsf{Designate}$ *is a polynomial-time Turing machine (PTM) which takes the public parameters, a message* m*, a signing public key* pks*, a signature* σ *on* m *for* pks *and a verifying public key as inputs and outputs a bit string.*
- $\Sigma.\mathsf{Fake}$ *is a PPTM which takes the public parameters, a message, a signing public key and a verifying secret key as inputs and outputs a bit string.*
- $\Sigma.\mathsf{DVerify}$ *is a deterministic PPTM which takes the public parameters, a message* m*, a bit string* τ*, a signing public key* pks*, a verifying public key* pkv *the matching verifying secret key* skv *as inputs. It outputs a bit. If the bit output is 1 then the bit string* τ *is said to be a* designated verifier signature *on* m *from* pks *to* pkv.

Σ *must satisfy the following properties, for all* $k \in \mathbb{N}$*, all* $\Upsilon \in \Sigma.\mathsf{Setup}[k]$*, all* $(\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathsf{SKeyGen}[\Upsilon]$*, all* $(\mathsf{pkv}, \mathsf{skv}) \in \Sigma.\mathsf{VKeyGen}[\Upsilon]$ *and all messages* m*:*

- Correctness of Signature:

$$\forall \sigma \in \Sigma.\mathsf{Sign}[\Upsilon, m, \mathsf{sks}], \quad \Sigma.\mathsf{Verify}[\Upsilon, m, \sigma, \mathsf{pks}] = \{1\}.$$

- Correctness of Designation:

$$\forall \sigma \in \Sigma.\mathsf{Sign}[\Upsilon, m, \mathsf{sks}], \quad \forall \tau \in \Sigma.\mathsf{Designate}[\Upsilon, m, \mathsf{pks}, \sigma, \mathsf{pkv}], \\ \Sigma.\mathsf{DVerify}[\Upsilon, m, \tau, \mathsf{pks}, \mathsf{pkv}, \mathsf{skv}] = \{1\}.$$

- Source Hiding:

$$\Sigma.\mathsf{Designate}(\Upsilon, m, \mathsf{pks}, \Sigma.\mathsf{Sign}(\Upsilon, m, \mathsf{sks}), \mathsf{pkv}]) = \Sigma.\mathsf{Fake}(\Upsilon, m, \mathsf{pks}, \mathsf{skv}).$$

The correctness properties insure that a properly formed (designated verifier) signature is always accepted by the (designated) verifying algorithm. The source

hiding property states that given a message m, a signing public key pks, a verifying public key pkv and a designated verifier signature τ on m from pks to pkv it is infeasible to determine if τ was produced by Σ.Designate or Σ.Fake.

The unforgeability notion is an extention of the classical notion of exitential unforgeability under a chosen-message attack as defined in [7].

Definition 2. *A universal designated verifier signature scheme is said strongly existentially unforgeable if no adversary (PPTM) $\mathcal{F}$ has a non-negligible advantage in the following game:*

1. *The challenger $\mathcal{C}$ takes as input a security parameter k and executes*
   $\Upsilon \leftarrow \mathit{UDVS}.\Sigma.\mathsf{Setup}(k)$,
   $(sk_S^\star, pk_S^\star) \leftarrow \mathit{UDVS}.\Sigma.\mathsf{KeyGen}(k, \Upsilon)$,
   $(sk_V^\star, pk_V^\star) \leftarrow \mathit{UDVS}.\mathsf{VKeyGen}(k, \Upsilon)$.
   It gives $pk_S^\star$ and $pk_V^\star$ to the forger $\mathcal{F}$ and keeps $sk_S^\star$ and $sk_V^\star$ to itself.
2. *The forger $\mathcal{F}$ can issue the following queries:*
   i) a signing query for some message m; the challenger $\mathcal{C}$ executes
   $$\sigma \leftarrow \mathit{UDVS}.\Sigma.\mathsf{Sign}(k, \Upsilon, m, sk_S^\star)$$
   and hands σ to $\mathcal{F}$;
   ii) a verification query for pairs $(m, \tilde{\sigma})$ of his choice; $\mathcal{C}$ returns to $\mathcal{F}$ the value $\mathit{UDVS}.\mathsf{DVerify}(k, \Upsilon, m, \tilde{\sigma}, pk, (sk_V^\star, pk_V^\star))$;
3. *$\mathcal{F}$ outputs a V-designated verifier signature $\tilde{\sigma}^\star$ for a message $m^\star$.*

The adversary $\mathcal{F}$ succeeds if $\mathit{UDVS}.\mathsf{DVerify}\big(k, \Upsilon, pk_S^\star, (sk_V^\star, pk_V^\star)\big) = 1$ and if $\tilde{\sigma}^\star$ has not been obtain from the signing oracle. An attacker $\mathcal{F}$ is said to $(\tau, q_s, q_v, \varepsilon)$-break the unforgeability of the UDVS scheme if he succeeds in the game within running time τ and with probability ε after having made q_s signing queries and q_v verification queries.

We will propose in Section 3 a new UDVS scheme compatible with standard Elgamal or DSA signatures, reaching these security notions. Another important security requirement concerning the undeniable signature familly is the notion of *anonymity* (roughly speaking, an anonymous designated verifier signature is indistinguishable from a random string.). Is has been precisely defined in [13] in terms of *privacy of signer's identity* for designated verifier signatures, but we will not need this property for our purpose.

2.3 Restricted Universal Designated Verifier Signatures

According to [8],

> *$\langle a \rangle$ restricted UDVS scheme is comprised of three procedures namely sign-up, designation and open. In the sign-up procedure, a user obtains a signature σ from the signer. This signature is publicly verifiable, but this is kept by the user (and hence, the signature holder) to be used in the designation procedure. $\langle \ldots \rangle$ In the designation procedure, the signature holder can designate the signature to up to t verifiers. The verifiers will be convinced with the authenticity of the signature, but they cannot convince any other third party about this fact. When the signature holder uses the signature to convince the $t+1$ verifier, the open procedure can be invoked to reveal the signature $\langle \ldots \rangle$.*

One of our purpose in this paper is to disprove the relevancy of this primitive. Therefore, it is necessary to describe formally this primitive.

Definition 3. *Let t be a positive integer. A* t-restricted universal designated verifier signature scheme Σ *is a 9-tuple*

$$\Sigma = (\mathit{Setup}, \mathit{SKeyGen}, \mathit{VKeyGen}, \mathit{Sign}, \mathit{Verify}, \mathit{Designate}, \mathit{Fake}, \mathit{DVerify}, \mathit{Open})$$

such that

- $(\mathit{Setup}, \mathit{SKeyGen}, \mathit{VKeyGen}, \mathit{Sign}, \mathit{Verify}, \mathit{Designate}, \mathit{Fake}, \mathit{DVerify})$ *is a universal designated signature scheme;*
- $\Sigma.\mathit{Open}$ *is a PPTM which takes the public parameters, a message,* $t+1$ *bit strings, a signing public key and* $t+1$ *verifying public keys and outputs a bit string.*

Σ *must satisfy the following property for all messages* m*:*

- Opening:

$$\begin{gathered}\forall k \in \mathbb{N}, \forall \Upsilon \in \Sigma.\mathit{Setup}[k], \forall (\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathit{SKeyGen}[\Upsilon], \forall \sigma \in \Sigma.\mathit{Sign}[\Upsilon, m, \mathsf{sks}], \\ \forall (\mathbf{pkv}, \mathbf{skv}) = [(\mathsf{pkv}_1, \mathsf{skv}_1), \ldots, (\mathsf{pkv}_{t+1}, \mathsf{skv}_{t+1})] \in \Sigma.\mathit{VKeyGen}[\Upsilon]^{t+1} \\ \forall \tau_1 \in \Sigma.\mathit{Designate}[\Upsilon, m, \mathsf{pks}, \sigma, \mathsf{pkv}_1], \\ \ldots \\ \forall \tau_{t+1} \in \Sigma.\mathit{Designate}[\Upsilon, m, \mathsf{pks}, \sigma, \mathsf{pkv}_{t+1}] \\ \#\{\tau_1, \ldots, \tau_{t+1}\} = t+1 \Rightarrow \Sigma.\mathit{Open}[\Upsilon, m, \tau_1, \ldots, \tau_{t+1}, \mathbf{pkv}, \mathsf{pks}] = \{\sigma\}.\end{gathered}$$

This property formalizes the (erroneous, as we will see) fact that when the signature is designated to convince $t+1$ verifiers, the open procedure can be invoked to reveal the original signature.

In [8], Huang *et al.* presented a restricted UDVS scheme based on BLS signatures.

2.4 Dedicated Signatures

The definition of dedicated signatures is not given explicitly and formally, but can be readily easily deduced from this informal description from [11]. We give here a formal definition (although the concept is pointless as we will see).

Definition 4. *A* dedicated signature scheme Σ *is an 7-tuple*

$$\Sigma = (\mathit{Setup}, \mathit{SKeyGen}, \mathit{VKeyGen}, \mathit{DDSCreate}, \mathit{Retrieve}, \mathit{Verify}, \mathit{Punish})$$

such that

- $\Sigma.\mathit{Setup}$ *is a PPTM which takes an integer* k *as input. The output are the* public parameters Υ*.* k *is called the* security parameter*.*

- $\Sigma.\mathit{SKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{sks}, \mathsf{pks})$ *where* sks *is called a* signing secret key *and* pks *a* signing public key*.*
- $\Sigma.\mathit{VKeyGen}$ *is a PPTM which takes the public parameters as input. The output is a pair* $(\mathsf{skv}, \mathsf{pkv})$ *where* skv *is called a* verifying secret key *and* pkv *a* verifying public key*.*
- $\Sigma.\mathit{DDSCreate}$ *is a PPTM which takes the public parameters, a message, a signing secret key and a verifying public key as inputs and outputs a bit string.*
- $\Sigma.\mathit{Retrieve}$ *is a PPTM which takes as input the public parameters, a message, a bit string, a signing public key and a verifying secret key as input. The output is a bitstring.*
- $\Sigma.\mathit{Verify}$ *is a PTM which takes the public parameters, a message* m*, a bit string* σ *and a signing public key* pks*. It outputs a bit. If the bit output is 1 then the bit string* σ *is said to be a* signature *on* m *for* pks*.*
- $\Sigma.\mathit{Punish}$ *is a PPTM which takes the public parameters, a message, two bit strings, a signing public key and a verifying public key and outputs a bit string.*

Σ must satisfy the following properties, for all $k \in \mathbb{N}$*, all* $\Upsilon \in \Sigma.\mathit{Setup}[k]$*, all* $(\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathit{SKeyGen}[\Upsilon]$*, all* $(\mathsf{pkv}, \mathsf{skv}) \in \Sigma.\mathit{VKeyGen}[\Upsilon]$ *and all messages* m*:*

- Correctness of Retrieval:

$$\forall \tau \in \Sigma.\mathit{DDSCreate}(\Upsilon, m, \mathsf{sks}, \mathsf{pkv}), \quad \forall \sigma \in \Sigma.\mathit{Retrieve}(\Upsilon, m, \tau, \mathsf{pks}, \mathsf{skv}), \\ \Sigma.\mathit{Verify}[\Upsilon, m, \sigma, \mathsf{pks}] = \{1\}.$$

- Punishement:

$$\forall \tau \in \Sigma.\mathit{DDSCreate}(\Upsilon, m, \mathsf{sks}, \mathsf{pkv}), \quad \forall \sigma \in \Sigma.\mathit{Retrieve}(\Upsilon, m, \tau, \mathsf{pks}, \mathsf{skv}), \\ \Sigma.\mathit{Punish}[\Upsilon, m, \tau, \sigma, \mathsf{pks}, \mathsf{pkv}] = \{\mathsf{skv}\}.$$

The first property can be restated as follows: $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is a signature scheme, where Sign is the PPTM which takes the public parameters, a message m, a signing key pair $(\mathsf{sks}, \mathsf{pks})$ and a verifying key pair $(\mathsf{skv}, \mathsf{pkv})$ as inputs, obtains τ by executing $\Sigma.\mathsf{DDSCreate}(\Upsilon, m, \mathsf{sks}, \mathsf{pkv})$ and returns σ obtained by executing $\Sigma.\mathsf{Retrieve}(\Upsilon, m, \tau, \mathsf{pks}, \mathsf{skv})$. The second property means that the verifier loses his own secrets when exhibiting the signature.

In [11], Klonowski *et al.* proposed a realization of this primitive based on Elgamal signatures. It is well known, that the original Elgamal signature scheme is existentially forgeable (see [17] for instance). As in [11], we consider only, in this paper, the variant of Elgamal scheme where a collision-resistant hash function is applied to the message to be signed and only the digest is actually "signed". This "hashed Elgamal signatures" has not been proven existentially unforgeable but there exist arguments in favor of their security in the so-called *generic group model* [2].

It is worth noting that this scheme presented is very similar to the anonymous designated verifier signature scheme proposed in 2003 by Saeednia, Kremer and Markowitch [18].

3 Universal Designated Verifier Elgamal and DSA Signatures

3.1 Description of the Scheme

We present in this section the first efficient universal designated verifier Elgamal and DSA signatures. We describe our new scheme with the Elgamal signature for simplicity. It works as follows:

- UDVS.Σ.Setup: public parameters include the output of a DL-parameter-generator as well as an integer n, a collision-resistant hash function $h : \{0,1\}^* \to \mathbb{Z}_p^*$: $\Upsilon := \{n, p, \mathbb{Z}_p^*, g, h\}$.
- UDVS.Σ.SKeyGen: a signer's private key is a randomly chosen $a \xleftarrow{R} \mathbb{Z}_p^*$; his public key consists of a group element $y_A = g^a \mod p$.
- UDVS.Σ.Sign: given a message $m \in \{0,1\}^*$, the signer picks $k \xleftarrow{R} \mathbb{Z}_p^*$ and sets $u = g^k \mod p$ and $v = k^{-1}(h(m) - au) \mod p-1$. The signature is $\sigma = (u, v)$.
- UDVS.Σ.Verify: a plain signature $\sigma = (u, v)$ on m is accepted if $y_A^u u^v = g^{h(m)} \mod p$.
- UDVS.VKeyGen : a designated verifier's private key is a random element $b \xleftarrow{R} \mathbb{Z}_p^*$; the matching public key is $y_B = g^b \mod p$.
- UDVS.Designate: the holder of a signature $\sigma = (u, v)$, who chooses B as designated verifier produces the designated verifier signature $\tau = (u, \mathcal{P})$ where $\mathcal{P}$ is the non-interactive zero-knowledge proof of knowledge of one discrete logarithm among two HVZKPK $\left(\log_u(g^{h(m)} y_A^{-u}) \vee \log_g(y_B)\right)$, derived from the interactive the proof described in Fig. 1.

  More precisely, the holder of the signature σ computes $e = h(c_1||c_2||m||u)$, with $y_1 = u$, $y_2 = g$, $\alpha_1 = g^{h(m)} y_A^{-u} \mod p$ and $\alpha_2 = y_B$. Then τ is the 5-tuple (u, e_1, e_2, s_1, s_2).
- UDVS.Fake: the designated verifier produces the designated verifier signature $\tau = (u, \mathcal{P})$ where u is a random element $u \xleftarrow{R} \mathbb{Z}_p$ and $\mathcal{P}$ is the non-interactive variant of the proof HVZKPK $\left(\log_u(g^{h(m)} y_A^{-u}) \vee \log_g(y_B)\right)$ (performed thanks to the knowledge of $\log_g(y_B)$).
- UDVS.DVerify: given a purported signature $\tau = (u, \mathcal{P})$, the designated verifier checks the correctness of the non-interactive proof $\mathcal{P} = (e_1, e_2, s_1, s_2)$ by computing $c_1 = (g^{h(m)} y_A^{-u})^{e_1} u^{s_1}$ and $c_2 = y_B^{e_2} g^{s_2}$ and checking that $h(c_1||c_2||m||u) = e_1 + e_2 \mod q$.

DSA signatures. The difference between Elgamal signatures and DSA signatures lies essentially in the fact that computation are done modulo a prime q dividing $p-1$ instead of working modulo $p-1$. Therefore, the technique previously described can be trivially adapted in the case of the DSA signatures. The resulting scheme is also the first universal designated verifier signature scheme based on DSA.

Remark 1. A UDVS scheme reaches the source-hiding property if and only if the designated verifier can produce one bitstring that is indistinguishable from

Prover		Verifier
$r_1, e_2, s_2 \in_R \mathbb{Z}_p^*$		
$c_1 = y_1^{r_1} \mod p$		
$c_2 = \alpha_2^{e_2} y_2^{s_2} \mod p$		
	$\xrightarrow{c_1, c_2}$	
		$e \in_R \mathbb{Z}_p^*$
	$\xleftarrow{e}$	
$e_1 = e_2 - e \mod p-1$		
$s_1 = r_1 - e_1 x_1 \mod p-1$		
	$\xrightarrow{(e_1, s_1), (e_2, s_2)}$	
		$c_1 \stackrel{?}{=} \alpha_1^{e_1} y_1^{s_1} \mod p$
		$c_2 \stackrel{?}{=} \alpha_2^{e_2} y_1^{s_2} \mod p$
		$e \stackrel{?}{=} e_1 + e_2 \mod p-1$

The prover knows $x_1 = \log_{y_1}(\alpha_1)$.

Fig. 1. HVZKPK $\left(\log_{y_1}(\alpha_1) \vee \log_{y_2}(\alpha_2)\right)$

one that was generated by the signature holder. In the Elgamal-based scheme, that we have described this property is fulfilled. However, it remains an open problem to design an efficient UDVS scheme compatible with Elgamal/DSA signatures with the stronger property that the designated verifier can always produce identically distributed transcripts that are indistinguishable from the one generated by the signature holder:

$$\forall k \in \mathbb{N}, \forall \Upsilon \in \Sigma.\mathsf{Setup}[k], \forall (\mathsf{pks}, \mathsf{sks}) \in \Sigma.\mathsf{SKeyGen}[\Upsilon],$$
$$\forall (\mathsf{pkv}, \mathsf{skv}) \in \Sigma.\mathsf{VKeyGen}[\Upsilon], \forall m \in \{0,1\}$$
$$\left(\tau \xleftarrow{R} \Sigma.\mathsf{Sign}(\Upsilon, m, \mathsf{sks}); \Sigma.\mathsf{Designate}(\Upsilon, m, \mathsf{pks}, \tau, \mathsf{pkv})\right) = \Sigma.\mathsf{Fake}(\Upsilon, m, \mathsf{pks}, \mathsf{skv}).$$

3.2 Security Analysis

Theorem 1 (Unforgeability). *Let $\mathcal{F}$ be a forger that $(t, q_s, q_v, \varepsilon)$-breaks the Elgamal UDVS scheme in the q_h-random oracle model, with $\varepsilon > 7q_H/2^k$. There exists a $(t', \lceil (14q_H + 2)q_s/\varepsilon \rceil, \varepsilon')$-EF-CMA algorithm $\mathcal{A}$ against hashed Elgamal signature such that $\varepsilon' \geq 1/9$ after running $\mathcal{F}$ by $\left\lceil \frac{2}{\varepsilon} + \frac{14q_h}{\varepsilon} \right\rceil$ times.*

Proof. The proof relies on the well-known forking technique proposed in 1996 by Pointcheval and Stern [17]: assuming that an attacker can forge a designated verifier signature, another algorithm could obtain, by replaying sufficiently many times this attacker with randomly chosen hash functions (i.e. random oracles), two forged designated verifier signatures of the same message and with the same randomness. Then, these two forged signatures could be used to solve some computational problem which is assumed to be intractable: producing an Elgamal forgery or computing a discrete logarithm.

The algorithm $\mathcal{A}$ takes as inputs the public parameters Υ, an Elgamal signing public-key $\mathsf{pks} = y$ and has access to a signing oracle $\mathfrak{S}_A$ that it can query up to $\lceil (14q_H+2)q_S/\varepsilon \rceil$ times. It produces a verifying public key pkv in such a way that if it obtains in the simulation the discrete logarithm of pkv in base g, then it will also know the discrete logarithm of pks in base g and therefore can readily forge a signature (*i. e.* $\mathsf{pkv} = y^r$ for a known $r \in [\![1, p-1]\!]$ picked uniformly at random). The adversary $\mathcal{A}$ executes the forger $\mathcal{F}$ many times (at most $\lceil (14q_h + 2)\varepsilon^{-1} \rceil$) on the entries $(\Upsilon, \mathsf{pks}, \mathsf{pkv})$ with different random tapes and/or random oracles. The algorithm $\mathcal{A}$ will run in two stages.

First stage: The forger $\mathcal{F}$, with random tape ϖ, can make q_S queries to the signing oracle $\mathfrak{S}_F$ and q_H queries to the random oracle $\mathcal{H}$. We denote by $(e_1, \ldots, e_{q_H})$ the list of the q_H answers of the random oracle. Therefore, we can see a random choice of the random oracle as a random choice of such a vector $\boldsymbol{e}$. In his first stage, $\mathcal{A}$ executes the forger $\mathcal{F}$ at most $(2/\varepsilon)$ times with different random tapes and random oracles, until it outputs a valid forgery. For each execution, the queries made by $\mathcal{F}$ to the signing oracle $\mathfrak{S}_F$ are simply transfered to $\mathfrak{S}_A$ and the returned signatures are stored by $\mathcal{A}$.

Eventually, in one execution $\mathcal{F}$ outputs a forged message/signature pair $(m^\star, \tau^\star)$ where $\tau^\star = (u^\star, e_1^\star, e_2^\star, s_1^\star, s_2^\star)$ under the public keys $(\mathsf{pks}, \mathsf{pkv})$. The probability that $\mathcal{F}$ outputs a valid forgery at the end of one execution is ε. Since $\mathcal{H}$ is a random oracle, the probability that $\mathcal{F}$ succeeds and has not submitted $(c_1^\star || c_2^\star || m^\star || u^\star)$ (where $c_1^\star = (g^{h(m)} y_A^{-u^\star})^{e_1^\star} u^{s_1^\star}$ and $c_2^\star = y_B^{e_2^\star} g^{s_2^\star}$) to $\mathcal{H}$ is less than 2^{-k}. Therefore, the probability that $\mathcal{F}$ returns a forged message/signature which has been queried to the random oracle in one execution is at least $6\varepsilon/7$. We define $\mathrm{Ind}(\varpi, \boldsymbol{e})$ to be the index of this query. We then define the sets

$$\mathcal{S}_i = \{(\varpi, \boldsymbol{e}) | \mathcal{F}^{\mathfrak{S}_F, \boldsymbol{e}}(\varpi) \text{ succeeds } \& \ \mathrm{Ind}(\varpi, \boldsymbol{e}) = i\}$$

for $i \in [\![1, q_H]\!]$ and $\mathcal{S} = \bigcup_{i=1}^{q_H} \mathcal{S}_i$. We have $\Pr[\mathcal{S}] \geq 6\varepsilon/7$.

Therefore, $\mathcal{A}$ at the end of the first stage will get at least one pair

$$(\varpi, \boldsymbol{e} = (e_1, \ldots, e_{q_H}))$$

(and a corresponding list of signatures) leading to a forgery with probability at least $1 - \exp(-12/7) \geq 4/5$ after having queried $\mathfrak{S}_A$ at most $\lceil 2q_S/\varepsilon \rceil$ times. Let us denote $\gamma = \mathrm{Ind}(\varpi, \boldsymbol{e})$ the index of query of the forged message to the random oracle. Let I be the set consisting of the most likely indices i:

$$I = \{i \in [\![1, q_H]\!], \Pr[\mathcal{S}_i | \mathcal{S}] \geq 1/2q_H\}.$$

It is easy to see [17, Lemma 3], that in case of success, the index γ lies in I with probability at least $1/2$. Moreover, with probability greater than $1/5$ the pair $(\varpi, \boldsymbol{e})$ belongs to S_γ.

Second stage: In the second stage, $\mathcal{A}$ executes $\mathcal{F}$ with the random tape ϖ which led to the forgery in the first step and different random oracles $\boldsymbol{e'} = (e'_1, \ldots, e'_{q_H})$ such that $e_i = e'_i$ for $i < \gamma$, e'_γ is picked uniformly at random in $[\![1, p-1]\!] \setminus \{e_\gamma\}$

and the e'_j's are picked uniformly at random in $[\![1, p-1]\!]$ for $j > \gamma$. $\mathcal{A}$ uses the signatures recorded in the first step to answer F signature queries before the γ-th random oracle query and its own signing oracle $\mathfrak{S}_A$ after that point.

After running $\mathcal{F}$ $\lceil 14q_H/\varepsilon \rceil$ times, $\mathcal{A}$ will obtain, if $(\varpi, \boldsymbol{e}) \in S_\gamma$, with probability at least 3/5 (see [17, Lemma 1] for instance), two valid designated verifier signatures on $m^\star$: $(u^\star, e_1^\star, e_2^\star, s_1^\star, s_2^\star)$ and $(u^\star, e_1^\dagger, e_2^\dagger, s_1^\dagger, s_2^\dagger)$ such that

$$e_1^\star + e_2^\star \neq e_1^\dagger + e_2^\dagger \mod p$$

and

$$(g^{h(m)} y_A^{-u^\star})^{e_1^\star} u^{s_1^\star} = (g^{h(m)} y_A^{-u^\star})^{e_1^\dagger} u^{s_1^\dagger} \mod p \text{ and } y_B^{e_2^\star} g^{s_2^\star} = y_B^{e_2^\dagger} g^{s_2^\dagger} \mod p.$$

If $e_1^\star \neq e_1^\dagger \mod p$, then $\mathcal{A}$ can readily obtain a valid Elgamal signature on $m^\star$ as $(u^\star, v^\star)$ where

$$v^\star = (s_1^\star - s_1^\dagger)/(e_1^\star - e_1^\dagger) \mod p.$$

Similarly, $e_2^\star \neq e_2^\dagger \mod p$, then $\mathcal{A}$ can retrieve the discrete logarithm of pkv in base g, and therefore the discrete logarithm of pks in base g and can easily produce a Elgamal signature on the message of his choice. □

The complexity assumption used in the previous theorem is the best we can hope for. Indeed, if there exists a polynomial time EF-CMA adversary against hashed Elgamal signature then there exists a forger that breaks the Elgamal UDVS scheme with the same advantage in the same running time.

The results from [16] show that it is very unlikely that the existential unforgeability of hashed Elgamal signatures can be reduced to the discrete logarithm problem in the standard security model. However, it is worth noting that the hashed Elgamal signature scheme is a special case of the protocol AbstractDSA which has been proposed and analyzed[1] by Brown in 2005 [2]. Let us recall the *Theorem 3* from [2] which asserts the existential unforgeability of the scheme in the generic group model:

Theorem 2 ([2]). *If there exists an $(\epsilon_F, \tau_F, q_F)$-forger $\mathcal{F}$ of AbstractDSA in the generic group model with uniform hash function h and an almost-invertible conversion function f in the generic group model for A_n, then there exists an (ϵ_C, τ_C)-collision-finder C_h and (ϵ_Z, τ_Z)-zero-finder where*

$$\epsilon_C + \epsilon_Z \geq \epsilon_F + \frac{\tau_F^2}{2n} \text{ and } \tau_C, \tau_Z \leq 2\tau_F.$$

For the specific instance of AbstractDSA investigated in this paper, the conversion function is the identity map which is trivially almost invertible [2, § 2.2.2] and therefore it can be argued that this theorem applies to hashed Elgamal signatures.

[1] We refer to [2] for the definitions of uniform hash function, almost-invertible conversion function and zero-finder adversary.

4 On the Weakness of Restricted Universal Designated Verifier and Dedicated Signatures

As mentioned above, the dedicated verifier may provide a zero-knowledge proof that it has a certain signature without presenting it. In many legal systems it would suffice in the court to present such a proof in order to derive legal consequences of the signed document.

The weakness and eventually the meaningless of these two concepts of signatures comes from the following theorem. It is among the most important results in zero-knowledge protocols and state the possible construction, using commitment schemes, of a zero-knowledge proof system for all languages in NP. This theorem was proved in [6] by Goldreich, Micali and Widgerson.

Theorem 3. *If non-uniform one-way functions exist, then there exists a computational zero-knowledge proof system of membership for all languages having an interactive proof system of membership.*

The impact of the theorem on the two kinds of signature is described below.

4.1 The Case of Restricted UDVS: Proving the Possession of a BLS Signature

In Huang *et al.* paper, the authors propose a pairing-based scheme. A user receive a BLS signature, and should convince only a given number of third parties of the validity of this signature. We argue that these restriction is not possible since the signature holder can always prove the validity of the BLS signature he holds by applying the zero-knowledge proof of the possession of a BLS signature described in Fig. 2.

Prover		Verifier
$r \in_R [\![1, q-1]\!]$ $w = e(P,P)^r$		
	$\xrightarrow{w}$	
		$c \in_R [\![0, q-1]\!]$
	$\xleftarrow{c}$	
$Y = rP + c\sigma$		
	$\xrightarrow{Y}$	
		$e(P,Y) \stackrel{?}{=} w \times e(aP, H(m))^c$

Prover: $(pk, sk) = (aP, a)$ and $\sigma = aH(m)$

Fig. 2. Proof of knowledge of a BLS signature [9]

4.2 The Case of Dedicated Signatures: Proving the Possession of an Elgamal Signature

In Klonowski *et al.* paper [11], the authors propose a scheme based on Elgamal signatures. In this scheme, the signer is suppose to construct his signature in such a way that the recipient cannot show the signature to third parties without being punished (for instance, his secret key is revealed). The sent signature is called a dedicated signature, but the recipient can derive a true Elgamal signature from it. Therefore once the holder has computed this Elgamal signature, he can use our new protocol defined in Section 3 to convince *any* third party without being punished. Indeed, the signature holder can transform the Elgamal signature into a signature designated to the third party.

5 Conclusion

Signatures of knowledge allow a prover to prove the knowledge of a secret with respect to some public information noninteractively. In our case, this helps to defeat some concepts which aim at controlling the holder of a signature. To this purpose, we propose for the two concepts of restricted universald designated verifier signatures and dedicated signatures two proofs which permit the signature holder to prove his knowledge of a signature without being troubled. The first one is a classical proof of a BLS signature. The second one is the first efficient designated verifier proof of an Elgamal signature.

An interesting open problem would be to have an efficient zero-knowledge proof of the possession of an Elgamal (or DSA) signature. What we proposed is an efficient designated verifier proof, which might not be relevant in some special cases. Moreover, the concept of punishing a signature holder who unauthorizedly disclose a signature is still open. The presence of an authority seems necessary to design such a scheme.

References

1. Boneh, D., Shacham, H., Lynn, B.: Short signatures from the Weil pairing. J. of Cryptology 17(4), 297–319 (2004)
2. Brown, D.R.L.: Generic Groups, Collision Resistance, and ECDSA. Des. Codes Cryptography 35(1), 119–152 (2005)
3. Chaum, D.: Designated Confirmer Signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 86–89. Springer, Heidelberg (1995)
4. Chaum, D., van Antwerpen, H.: Undeniable Signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
5. Elgamal, T.: A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
6. Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. J. ACM 38(3), 691–729 (1991)

7. Goldwasser, S., Micali, S., Rives, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 1(2), 281–308 (1988)
8. Huang, X., Susilo, W., Mu, Y., Zhang, F.: Restricted Universal Designated Verifier Signature. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 874–882. Springer, Heidelberg (2006)
9. Hufschmitt, E., Lefranc, D., Sibert, H.: A Zero-Knowledge Identification Scheme in Gap Diffie-Hellman Groups (preprint, 2005)
10. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
11. Klonowski, M., Kubiak, P., Kutylowski, M., Lauks, A.: How to Protect a Signature from Being Shown to a Third Party. In: Fischer-Hübner, S., Furnell, S., Lambrinoudakis, C. (eds.) TrustBus 2006. LNCS, vol. 4083, pp. 192–202. Springer, Heidelberg (2006)
12. Laguillaumie, F., Libert, B., Quisquater, J.-J.: Universal Designated Verifier Signatures Without Random Oracles or Non-Black Box Assumptions. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 63–77. Springer, Heidelberg (2006)
13. Laguillaumie, F., Vergnaud, D.: Designated Verifier Signatures: Anonymity and Efficient Construction from *any* Bilinear Map. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 107–121. Springer, Heidelberg (2005)
14. Lipmaa, H., Wang, G., Bao, F.: Designated Verifier Signature Schemes: Attacks, New Security Notions and A New Construction. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 61–71. Springer, Heidelberg (2005)
15. Nguyen, K.Q., Bao, F., Mu, Y., Varadharajan, V.: Zero-Knowledge Proofs of Possession of Digital Signatures and its Applications. In: Varadharajan, V., Mu, Y. (eds.) Information and Communication Security. LNCS, vol. 1726, pp. 103–118. Springer, Heidelberg (1999)
16. Paillier, P., Vergnaud, D.: Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (2005)
17. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. J. Cryptology 13(3), 361–396 (2000)
18. Saeednia, S., Kremer, S., Markowitch, O.: An Efficient Strong Designated Verifier Signature Scheme. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 40–54. Springer, Heidelberg (2004)
19. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal Designated-Verifier Signatures. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
20. Steinfeld, R., Wang, H., Pieprzyk, J.: Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 86–100. Springer, Heidelberg (2004)
21. Vergnaud, D.: New Extensions of Pairing-based Short Signatures into Universal Designated Verifier Signatures. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 58–69. Springer, Heidelberg (2006)

Identity-Based Proxy Re-encryption Without Random Oracles*

Cheng-Kang Chu and Wen-Guey Tzeng

Department of Computer Science, National Chiao Tung University,
Hsinchu, Taiwan 300
{ckchu,wgtzeng}@cs.nctu.edu.tw

Abstract. A proxy re-encryption scheme allows Alice to temporarily delegate the decryption rights to Bob via a proxy. Alice gives the proxy a re-encryption key so that the proxy can convert a ciphertext for Alice into the ciphertext for Bob. In this paper, we propose two identity-based proxy re-encryption schemes, which are both proved secure in the standard model. The first one is efficient in both computation and ciphertext length, and the other one achieves chosen-ciphertext security. Our solutions answer the open problems left in the previous work.

Keywords: Proxy re-encryption, identity-based encryption, standard model.

1 Introduction

A proxy re-encryption (PRE) scheme involves three parties: Alice, Bob, and a proxy. Alice gives the proxy a re-encryption key so that the proxy can convert the ciphertext encrypted under Alice's public-key into the ciphertext for Bob. Certainly, the proxy cannot learn the plaintext or the secret keys of Alice or Bob. By using PRE, Alice can temporarily forward the ciphertext to Bob without revealing her secret key. A natural application of PRE is the re-encryption of e-mails: when Alice takes a leave of absence, she can let Bob read her encrypted e-mails. Once Alice comes back, the proxy stops transferring the e-mails.

Green and Ateniese [10] proposed the first *identity-based* PRE (IB-PRE). It allows the proxy to convert an encryption under Alice's identity into the encryption under Bob's identity. So Alice can assign the re-encryption key to a proxy with Bob's identity only. Their schemes are based on Boneh and Franklin's identity-based encryption (IBE) scheme [5], which was shown to be secure in the random oracle model. However, many researchers have expressed doubts about the random oracle heuristic. For example, some uninstantiable random oracle cryptosystems, which are secure in the random oracle model but are insecure in any real world implementation, were proposed [7,2]. Therefore, it is natural to ask whether secure IB-PRE scheme exists in the standard model, i.e., without random oracles.

* Research supported in part by Taiwan Information Security Center at NCTU (TWISC@NCTU), and National Science Council project NSC 95-2221-E-009-030.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 189–202, 2007.

In this paper, we propose two IB-PRE schemes, which are both proved secure in the standard model. The first one is efficient in both computation and ciphertext length, and the other one achieves chosen-ciphertext security. Both of our schemes satisfy the following properties of PRE, which are mentioned in [1,10].

- Unidirectionality. Alice can delegate decryption rights to Bob without permitting she to decrypt Bob's ciphertext.
- Non-Interactivity. Alice can compute re-encryption keys without the participation of Bob or the private key generator (PKG).
- Multi-Use. The proxy can re-encrypt a ciphertext multiple times, e.g. re-encrypt from Alice to Bob, and then re-encrypt the result from Bob to Carol.

The schemes proposed by Green and Ateniese [10] also satisfy these properties except that their CCA-secure construction is not multi-use. Therefore, we give the answers to the two open problems left in [10] by providing

1. IB-PRE schemes secure in the standard model; and
2. a multi-use CCA-secure IB-PRE scheme.

Related Works. Mambo and Okamoto [14] first introduced the notion of PRE. They gave some transformations that allow the original recipient to forward specific ciphertexts to another recipient. Blaze et al. [3] later provided another definition for PRE that allows the keyholder to publish the proxy function and have it applied by untrusted parties without further involvement by the original keyholder. After that, several public-key based PRE were continuously proposed [13,12,1,11,9]. Finally, Green and Ateniese [10] provided identity-based PRE. Note that Ivan and Dodis [12] also proposed an identity-based PRE scheme in which the PKG delegates decryption rights for *all* identities in the system. Therefore, individual users cannot delegate their decryption rights. The concept of their construction is much different from the approach of Green and Ateniese.

Organizations. In the rest of this paper we first give some preliminaries, including the IBE schemes without random oracles (Section 2) and the definition of IB-PRE (Section 3). Then we present a CPA-secure IB-PRE scheme in Section 4 and a CCA-secure IB-PRE scheme in Section 5. The summary of this paper is provided in Section 6.

2 Backgrounds

We briefly describe the groups and IBE schemes that will be used in our constructions.

2.1 Pairings

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two (multiplicatively) cyclic groups of prime order p. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is a map with the following properties:

– Bilinear: for all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
– Non-degenerate: for some $g \in \mathbb{G}$, $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_1$, and the bilinear map are efficiently computable.

2.2 Waters IBE Scheme

Waters [15] introduces an efficient IBE scheme without random oracles (Wa-IBE). The scheme is described as follows.

– **Setup**(1^λ)**:** On input security parameter 1^λ, randomly choose two groups $\mathbb{G}$ and $\mathbb{G}_1$ with prime order p, a bilinear map e and a generator g defined above. Let $\alpha \in \mathbb{Z}_p$ be a randomly chosen secret. Set the value $g_1 = g^\alpha$ and choose the value $g_2 \in \mathbb{G}$ randomly. Let v be an n-bit string and $\mathcal{V}$ be the set of all i for which the i-th bit of v is one. Define $F(v) = u' \prod_{i \in \mathcal{V}} u_i$, where $u', u_1, u_2, \ldots, u_n$ are chosen at random from $\mathbb{G}$. The public parameter $\mu = (g, g_1, g_2, F(\cdot))$ and the master secret key $mk = g_2^\alpha$ are outputted.
– **KeyGen**(μ, mk, v)**:** Let v be an n-bit string representing an identity. The private key for v is computed as

$$d_v = (g_2^\alpha F(v)^r, g^r),$$

where $r \in_R \mathbb{Z}_p$.
– **Encrypt**(μ, v, M)**:** For the message M and identity v, compute the ciphertext as

$$C = (M \cdot e(g_1, g_2)^t, g^t, F(v)^t),$$

where $t \in_R \mathbb{Z}_p$.
– **Decrypt**(μ, d, C)**:**
Let $d = (d_1, d_2)$ and $C = (c_1, c_2, c_3)$. Compute the message as

$$M = c_1 \frac{e(d_2, c_3)}{e(d_1, c_2)}.$$

By the security argument in [15], we have the following theorem.

Theorem 1. *Wa-IBE is semantically secure assuming the decisional BDH assumption holds.*

2.3 Waters CCA-Secure IBE Scheme

The Wa-IBE scheme can be naturally extended to a hierarchical IBE scheme (Wa-HIBE). Moreover, by the results of Canetti et al. [8], further improved upon by Boneh and Katz [6], one can build a CCA-secure IBE scheme from a 2-level HIBE scheme. As stated in [15], one can use Wa-IBE at the first level and the scheme of Boneh and Boyen [4] with only selective-ID security at the second level to get a more efficient CCA-secure IBE scheme. However, for brevity, we

directly use the 2-level Wa-HIBE to construct a CCA-secure IBE scheme (Wa-CCA-IBE).

The following scheme is a CCA-secure IBE scheme from Wa-HIBE [15] and the results of Boneh and Katz [6].

– **Setup**(1^λ)**:** On input security parameter 1^λ, the parameters are chosen like the Wa-IBE scheme except that the function F is replaced by two functions

$$F_1(v) = u_1' \prod_{i \in \mathcal{V}} u_{1,i} \qquad \text{and} \qquad F_2(w) = u_2' u_{2,0} \prod_{i \in \mathcal{W}} u_{2,i},$$

where $u_1', u_{1,1}, \ldots, u_{1,n}, u_2', u_{2,0}, u_{2,1}, \ldots, u_{2,n}$ are chosen at random from $\mathbb{G}$, v, w are two n-bit strings and $\mathcal{V}, \mathcal{W}$ are the set of all i for which the i-th bit of v and w is one, respectively. Moreover, let $(\mathcal{G}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme in which the verification key output by $\mathcal{G}(1^{\lambda'})$ has length n. The public parameter

$$\mu = (g, g_1, g_2, F_1(\cdot), F_2(\cdot), (\mathcal{G}, \mathsf{Sign}, \mathsf{Vrfy}))$$

and the master secret key $mk = g_2^\alpha$ are outputted.

– **KeyGen**(μ, mk, v)**:** Let v be an n-bit string representing an identity. Then the private key for v is computed as

$$d_v = (g_2^\alpha F_1(v)^r, g^r),$$

where $r \in_R \mathbb{Z}_p$.

– **Encrypt**(μ, v, M)**:** Perform $\mathcal{G}(1^{\lambda'})$ to get (vk, sk). For the message M and identity v, compute the ciphertext as

$$\tilde{C} = (M \cdot e(g_1, g_2)^t, g^t, F_1(v)^t, F_2(vk)^t),$$

where $t \in_R \mathbb{Z}_p$. Moreover, compute $\sigma = \mathsf{Sign}_{sk}(\tilde{C})$. Output the ciphertext $C = (\tilde{C}, vk, \sigma)$.

– **Decrypt**(μ, d, C)**:** Let $d = (d_1, d_2)$ and $C = (c_1, c_2, c_3, c_4, vk, \sigma)$. Check if

$$\mathsf{Vrfy}_{vk}((c_1, c_2, c_3, c_4), \sigma) \stackrel{?}{=} 1.$$

If not, output $\perp$. Otherwise, compute $d_1' = d_1 F_2(vk)^{r'}$ and $d_2' = g^{r'}$, where $r' \in_R \mathbb{Z}_p$. Then decrypt C as

$$M = c_1 \frac{e(d_2, c_3) e(d_2', c_4)}{e(d_1', c_2)}.$$

By the security argument in [6], we have the following theorem.

Theorem 2. *If Wa-HIBE is secure against chosen-plaintext attacks and* $(\mathcal{G}, \mathsf{Sig}, \mathsf{Vrfy})$ *is a one-time signature scheme, then Wa-CCA-IBE is secure against chosen-ciphertext attacks.*

3 Definitions

We give the definition of a secure IB-PRE scheme in this section.

3.1 Identity-Based Proxy Encryption

In addition to the four algorithms of an IBE scheme, an IB-PRE scheme needs two other algorithms: **RKGen** and **Reencrypt** to generate re-encryption key and re-encrypt ciphertexts, respectively.

Definition 1 ([10]). *An identity-based proxy re-encryption scheme consists of algorithms:*

- ***Setup***(1^λ)**:** *On input a security parameter, the public parameter μ and master secret key mk are outputted.*
- ***KeyGen***(μ, mk, v)**:** *On input the master secret key mk and an identity v, output the decryption key d_v.*
- ***Encrypt***(μ, v, m)**:** *On input an identity v and a message m, output the ciphertext C_v.*
- ***RKGen***(μ, d_{v_1}, v_1, v_2)**:** *On input a decryption key d_{v_1} and identities v_1, v_2, output the re-encryption key $d_{v_1 \to v_2}$.*
- ***Reencrypt***$(\mu, d_{v_1 \to v_2}, C_{v_1})$**:** *On input a re-encryption key $d_{v_1 \to v_2}$ and a ciphertext C_{v_1}, output the re-encrypted ciphertext C_{v_2}.*
- ***Decrypt***(μ, d_v, C_v)**:** *On input a private key d_v and a ciphertext C_v, output the plaintext m or $\perp$.*

Correctness. *Suppose $(\mu, mk) \leftarrow$ **Setup**(1^λ) and $d_v \leftarrow$ **KeyGen**(μ, mk, v). Let C_v be a ciphertext output from*

1. ***Encrypt***(μ, v, m)*; or*
2. ***Reencrypt***$(\mu, d_{v' \to v}, C_{v'})$,
   *where $d_{v' \to v} \leftarrow$ **RKGen**$(\mu, d_{v'}, v', v), d_{v'} \leftarrow$ **KeyGen**(μ, mk, v'), and $C_{v'} \leftarrow$ **Encrypt**(μ, v', m),*

for any identity v. Then the following equation holds:

$$m = \textbf{\textit{Decrypt}}(\mu, d_v, C_v).$$

3.2 Security

Now we define the security game of an IB-PRE scheme. Note that we assume the proxy will not collude with the re-encryption recipient. This is reasonable because given the re-encryption key, the collusion of these two parties can re-encrypt ciphertexts and decrypt them naturally.

Definition 2. *The security of an IB-PRE scheme is defined according to the following game $Exp^{\mathcal{A},\mathsf{IND\text{-}PrID\text{-}ATK}}$, where $ATK \in \{CPA, CCA\}$.*

1. *Setup.* Perform $\boldsymbol{Setup}(1^\lambda)$ to get (μ, mk) and give μ to $\mathcal{A}$.
2. *Query phase 1.* $\mathcal{A}$ makes the following queries.
   (a) $\textsc{Extract}(v)$: return $d_v = \boldsymbol{KeyGen}(\mu, mk, v)$ to $\mathcal{A}$.
   (b) $\textsc{RKExtract}(v_1, v_2)$: return $d_{v_1 \to v_2} = \boldsymbol{RKGen}(\mu, d_{v_1}, v_1, v_2)$ to $\mathcal{A}$, where $d_{v_1} = \boldsymbol{KeyGen}(\mu, mk, v_1)$.
   If ATK=CCA, $\mathcal{A}$ can make the additional queries:
   (c) $\textsc{Reencrypt}(v_1, v_2, C_{v_1})$: return $C_{v_2} = \boldsymbol{Reencrypt}(\mu, d_{v_1 \to v_2}, v_1, v_2, C_{v_1})$ to $\mathcal{A}$, where $d_{v_1 \to v_2} = \boldsymbol{RKGen}(\mu, d_{v_1}, v_1, v_2)$, $d_{v_1} = \boldsymbol{KeyGen}(\mu, mk, v_1)$.
   (d) $\textsc{Decrypt}(v, C_v)$: return $M = \boldsymbol{Decrypt}(\mu, d_v, C_v)$ to $\mathcal{A}$, where $d_v = \boldsymbol{KeyGen}(\mu, mk, v)$.
3. *Challenge.* $\mathcal{A}$ presents (v^*, m_0, m_1). If the queries
   – $\textsc{Extract}(v^*)$; and
   – $\textsc{RKExtract}(v^*, v')$ and $\textsc{Extract}(v')$ for any identity v',
   are never made, return $C^* = \boldsymbol{Encrypt}(\mu, v^*, m_b)$ to $\mathcal{A}$, where $b \in_R \{0, 1\}$.
4. *Query phase 2.* $\mathcal{A}$ continues making queries as in the Query phase 1, except for the following queries
   – $\textsc{Extract}(v^*)$;
   – $\textsc{RKExtract}(v^*, v')$ and $\textsc{Extract}(v')$ for any identity v';
   – $\textsc{RKExtract}(v^*, v')$ and $\textsc{Decrypt}(v', C_{v'})$ for any identity v' and any ciphertext $C_{v'}$.
   – $\textsc{Reencrypt}(v^*, v', C^*)$ and $\textsc{Extract}(v')$ for any identity v';
   – $\textsc{Decrypt}(v^*, C_{v^*})$; and
   – $\textsc{Decrypt}(v', C_{v'})$ for any identity v', where $C_{v'} \leftarrow \textsc{Reencrypt}(v^*, v', C^*)$.
5. *Guess.* $\mathcal{A}$ outputs the guess $b' \in \{0, 1\}$.

If $b' = b$, $\mathcal{A}$ wins the game. Let $\mathcal{O}^{ATK}$ be the set of oracles that $\mathcal{A}$ can query under ATK=CPA or CCA, and $\tilde{\mathcal{O}}^{ATK}$ be the set of the same oracles with the restrictions in Query phase 2. Then the advantage of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the above game is defined as:

$$\begin{aligned} Adv_{\mathcal{A}}^{\text{IND-PrID-ATK}} = \Pr[b' = b : b \in_R \{0,1\}, (\mu, mk) \leftarrow \boldsymbol{Setup}(1^\lambda), \\ (v^*, m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}^{ATK}}(\mu), C^* \leftarrow \boldsymbol{Encrypt}(\mu, v^*, m_b), \\ b' \leftarrow \mathcal{A}_2^{\tilde{\mathcal{O}}^{ATK}}(st, C^*)] - \frac{1}{2}. \end{aligned}$$

We say that an IB-PRE scheme is IND-PrID-ATK secure, ATK∈{CPA,CCA}, if for all probabilistic polynomial time algorithm $\mathcal{A}$ and negligible function ϵ,

$$Adv_{\mathcal{A}}^{\text{IND-PrID-ATK}} \leq \epsilon(k).$$

4 The First Construction

We present the first scheme IB-PRE-I in this section. The scheme is based on Wa-IBE, and provides CPA security. It is efficient in both computation and ciphertext length.

4.1 The Scheme IB-PRE-I

For brevity, we first assume the proxy re-encrypts ciphertexts once. The multi-use property will be discussed later.

- **Setup**(1^λ)**:** On input security parameter 1^λ, randomly choose two groups $\mathbb{G}$ and $\mathbb{G}_1$ with prime order p, a bilinear map e and a generator g defined above. Let $l \leq |p| - 1$ and $E_1 : \{0,1\}^l \to \mathbb{G}_1, E_2 : \{0,1\}^l \to \mathbb{G}$ be two encodings. Let $\alpha \in \mathbb{Z}_p$ be a randomly chosen secret. Set the value $g_1 = g^\alpha$ and choose the value $g_2 \in \mathbb{G}$ randomly. Let v be an n-bit string and $\mathcal{V}$ be the set of all i for which the i-th bit of v is one. Define $F(v) = u' \prod_{i \in \mathcal{V}} u_i$, where $u', u_1, u_2, \ldots, u_n$ are chosen at random from $\mathbb{G}$. The public parameter $\mu = (g, g_1, g_2, F(\cdot), E_1(\cdot), E_2(\cdot))$ and the master secret key $mk = g_2^\alpha$ are outputted.
- **KeyGen**(μ, mk, v)**:** Let v be an n-bit string representing an identity. The private key for v is computed as

$$d_v = (g_2^\alpha F(v)^r, g^r),$$

  where $r \in_R \mathbb{Z}_p$.
- **Encrypt**(μ, v, m)**:** For the message $m \in \{0,1\}^l$ and identity v, compute the ciphertext as

$$C_v = (M \cdot e(g_1, g_2)^t, g^t, F(v)^t),$$

  where $t \in_R \mathbb{Z}_p$ and $M = E_1(m)$.
- **RKGen**(μ, d_{v_1}, v_1, v_2)**:** Let $d_{v_1} = (d_1, d_2)$. Compute the re-encryption key for v_2 as

$$d_{v_1 \to v_2} = (d_1 K^{-1}, d_2, R),$$

  where $k \in_R \{0,1\}^l, K = E_2(k) \in \mathbb{G}$ and $R \leftarrow$ **Encrypt**(μ, v_2, k).
- **Reencrypt**$(\mu, d_{v_1 \to v_2}, C_{v_1})$**:** Let $d_{v_1 \to v_2} = (\hat{d}_1, d_2, R)$ and $C_{v_1} = (c_1, c_2, c_3)$. Re-encrypt the ciphertext:

$$C_{v_2} = (c_1 \frac{e(d_2, c_3)}{e(\hat{d}_1, c_2)}, c_2, R)$$

- **Decrypt**(μ, d_v, C_v)**:**
  If C_v is a regular encryption, let $d_v = (d_1, d_2)$ and $C_v = (c_1, c_2, c_3)$. Decrypt C_v as usual:

$$M = c_1 \frac{e(d_2, c_3)}{e(d_1, c_2)}.$$

  If C_v is a re-encrypted ciphertext, let $C_v = (c_1, c_2, R)$. Decrypt C_v by performing

$$k \leftarrow \textbf{Decrypt}(\mu, d_v, R), \quad K = E_2(k), \quad \text{and} \quad M = c_1 / e(c_2, K).$$

  Output $m = E_1^{-1}(M)$.

We can see that the recipient only needs k to decrypt the re-encrypted ciphertext. Therefore the proxy can further convert the re-encrypted ciphertext to others by iteratively re-encrypting R. As long as the recipient can decrypt the ciphertext to get k, the message m can be computed as well. The scheme satisfies the multi-use property. Note that R is not a single group element, but we can just encrypt all elements in the tuple. The cost of ciphertext grows linearly with the number of re-encryptions. As stated in [10], it seems to be inevitable for a non-interactive scheme.

4.2 Security

Next we prove that the scheme IB-PRE-I is IND-PrID-CPA secure if the Wa-IBE scheme is IND-ID-CPA secure. For the adversary $\mathcal{A}$ breaking IB-PRE-I, we build a simulator $\mathcal{B}$ to break Wa-IBE. The simulator maintains a table with tuples (β, v_1, v_2), where $\beta \in \{0, 1\}$ and v_1, v_2 are two identities. Since we have to query key extraction oracle of Wa-IBE for v to answer $\mathcal{A}$'s queries to RKExtract$(v, \cdot)$. If $\mathcal{A}$ queries RKExtract$(v, \cdot)$ and later sends v as the challenge identity, then $\mathcal{B}$ cannot get the challenge ciphertext of Wa-IBE for v and the simulation must fail. Therefore when $\mathcal{A}$ queries RKExtract$(v, \cdot)$, $\mathcal{B}$ either sends v to the key extraction oracle of Wa-IBE or randomly generates a re-encryption key. We set $\beta = 1$ if $\mathcal{B}$ returns the correct key and $\beta = 0$ otherwise.

Theorem 3. *Suppose there is an adversary $\mathcal{A}$ that has advantage ϵ against the game $Exp^{\mathcal{A},\mathsf{IND\text{-}PrID\text{-}CPA}}$. Then there is an algorithm $\mathcal{B}$ that breaks Wa-IBE with advantage at least*

$$Adv_{\mathcal{B}}^{\mathsf{IND\text{-}ID\text{-}CPA}} \geq \epsilon / e(1 + q_E),$$

where q_E is the maximal number of $\mathcal{A}$'s queries to Extract*, and e is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.*

Proof. Assume that there is an adversary $\mathcal{A}$ breaking IB-PRE-I. We construct an algorithm $\mathcal{B}$ to break Wa-IBE. Given the public parameter μ of Wa-IBE scheme, $\mathcal{B}$ performs the following steps. Note that $\mathcal{B}$ maintains a table with tuples $(\beta, v_1, v_2) \in \{0, 1\} \times \{0, 1\}^n \times \{0, 1\}^n$. Let $*$ denote the wildcard. Without loss of generality, we assume an input is queried to an oracle only once.

1. Setup. Give the parameter μ to $\mathcal{A}$.
2. Query phase 1. $\mathcal{A}$ can make the following queries.
   (a) Extract(v): $\mathcal{B}$ first generates a random coin β so that $\Pr[\beta = 1] = \delta$ for some δ that will be determined later. If $\beta = 0$, or $(0, v, *)$ or $(0, *, v)$ already exists on the table, $\mathcal{B}$ outputs a random bit and aborts. Otherwise, $\mathcal{B}$ sends the query to the key extraction oracle of Wa-IBE to get d_v, and returns d_v to $\mathcal{A}$. $\mathcal{B}$ also records $(1, v, v)$ on the table.
   (b) RKExtract(v_1, v_2): $\mathcal{B}$ chooses a random coin β as in the Extract. If $\beta = 1$, or $(1, v_1, v_1)$ or $(1, v_2, v_2)$ already exists on the table, $\mathcal{B}$ queries the key extraction oracle of Wa-IBE for v_1, and then computes the re-encryption key as the original scheme and returns it to $\mathcal{A}$. Otherwise, $\mathcal{B}$

returns a random re-encryption key $d_{v_1 \to v_2} = (x, y, \mathbf{Encrypt}(\mu, v_2, z))$ for $x, y \in_R \mathbb{G}$ and $z \in_R \{0,1\}^l$. Finally, $\mathcal{B}$ records the tuple (β, v_1, v_2) on the table.

3. Challenge. $\mathcal{A}$ sends (v^*, m_0, m_1) to $\mathcal{B}$, if $(1, v^*, v')$ exists on the table for any v', $\mathcal{B}$ randomly outputs a bit and aborts. Otherwise, $\mathcal{B}$ sends the same challenge (v^*, m_0, m_1) to the challenger of Wa-IBE. When the challenger returns ciphertext C^*, $\mathcal{B}$ returns C^* to $\mathcal{A}$.
4. Query phase 2. $\mathcal{A}$ continues making the following queries, except for the restrictions described in Definition 2.
   (a) EXTRACT(v): $\mathcal{B}$ answers queries as in the Query phase 1.
   (b) RKEXTRACT(v_1, v_2): For all $v_1 \neq v^*$, $\mathcal{B}$ queries the key extraction oracle of Wa-IBE for v, computes the re-encryption key as the original scheme and returns it to $\mathcal{A}$. For $v_1 = v^*$, $\mathcal{B}$ returns a random re-encryption key $d_{v_1 \to v_2} = (x, y, \mathbf{Encrypt}(\mu, v_2, z))$ for $x, y \in_R \mathbb{G}$ and $z \in_R \{0,1\}^l$, and records the tuple $(0, v_1, v_2)$ on the table.
5. Guess. When $\mathcal{A}$ outputs the guess b', $\mathcal{B}$ outputs b'.

We can see that if $\mathcal{B}$ does not abort during the game, the view of $\mathcal{A}$ is identical to the real attack except for some incorrect re-encryption keys (when $\beta = 0$). We will address this case later (in Lemma 1) by showing that $\mathcal{A}$ cannot distinguish these random generated keys from the real keys. So now we only need to calculate the probability that $\mathcal{B}$ aborts during the game. Suppose $\mathcal{A}$ makes a total of q_E private key extraction queries. The probability that $\mathcal{B}$ does not abort in phases 1 or 2 is δ^{q_E}. The probability that it does not abort during the challenge step is $1 - \delta$. Therefore, the probability that $\mathcal{B}$ does not abort during the game is $\delta^{q_E}(1 - \delta)$. This value is maximized at $\delta_{opt} = 1 - 1/(q_E + 1)$. Using δ_{opt}, the probability that $\mathcal{B}$ does not abort is at least $1/e(1 + q_E)$. So $\mathcal{B}$'s advantage is at least $\epsilon/e(1 + q_E)$. □

It remains to show that no adversary $\mathcal{A}$ can distinguish the simulation from a real interaction in which all values have the correct form. We complete this part by the following lemma.

Lemma 1 (Indistinguishability of simulations). *If Wa-IBE is IND-ID-CPA secure, then the simulation in the proof of Theorem 3 is computationally indistinguishable from the real scheme.*

Proof. The simulation in the proof of Theorem 3 almost acts the same as the real scheme, except for the incorrect form of re-encryption keys for $\beta = 0$. Therefore we only consider the indistinguishability of randomly chosen values $(x, y, \mathbf{Encrypt}(\mu, v_2, z))$ and the real re-encryption key. Since (x, y) must be a valid form of $(d_1 K^{-1}, d_2)$ for some valid private key (d_1, d_2) of v and $K \in \mathbb{G}$, the problem is equivalent to the distinguishability of the encryption of $z \in_R \{0,1\}^l$ and the encryption of k. Therefore the simulation works if Wa-IBE is IND-ID-CPA secure. □

5 The Chosen-Ciphertext Secure Construction

The CCA-secure IB-PRE scheme is presented in this section. It is based on Wa-CCA-IBE.

5.1 The Scheme IB-PRE-II

Like IB-PRE-I, we first assume the proxy re-encrypts ciphertexts once.

- **Setup**(1^λ)**:** On input security parameter 1^λ, the parameters are chosen like the first construction except that the function F is replaced by two functions

$$F_1(v) = u'_1 \prod_{i \in \mathcal{V}} u_{1,i} \qquad \text{and} \qquad F_2(w) = u'_2 u_{2,0} \prod_{i \in \mathcal{W}} u_{2,i},$$

  where $u'_1, u_{1,1}, \ldots, u_{1,n}, u'_2, u_{2,0}, u_{2,1}, \ldots, u_{2,n}$ are chosen at random from $\mathbb{G}$, v, w are two n-bit strings and $\mathcal{V}, \mathcal{W}$ are the set of all i for which the i-th bit of v and w is one, respectively. Let $l \leq |p| - 2$ and $E_1 : \{0,1\}^{l+1} \to \mathbb{G}_1, E_2 : \{0,1\}^l \to \mathbb{G}$ be two encodings. Moreover, let $(\mathcal{G}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme in which the verification key output by $\mathcal{G}(1^{\lambda'})$ has length n. The public parameter

$$\mu = (g, g_1, g_2, F_1(\cdot), F_2(\cdot), (\mathcal{G}, \mathsf{Sign}, \mathsf{Vrfy}))$$

  and the master secret key $mk = g_2^\alpha$ are outputted.
- **KeyGen**(μ, mk, v)**:** Let v be an n-bit string representing an identity. Then the private key for v is computed as

$$d_v = (g_2^\alpha F_1(v)^r, g^r),$$

  where $r \in_R \mathbb{Z}_p$.
- **Encrypt**(μ, v, m)**:** Perform $\mathcal{G}(1^{\lambda'})$ to get (vk, sk). For the message $m \in \{0,1\}^l$ and identity v, compute

$$\tilde{C} = (M \cdot e(g_1, g_2)^t, g^t, F_1(v)^t, F_2(vk)^t),$$

  where $t \in_R \mathbb{Z}_p$ and $M = E_1(m||0)$. Moreover, compute $\sigma = \mathsf{Sign}_{sk}(\tilde{C})$. Output the ciphertext $C_v = (\tilde{C}, vk, \sigma)$.
- **RKGen**(μ, d_{v_1}, v_1, v_2)**:** Let $d_{v_1} = (d_1, d_2)$. Compute the re-encryption key for v_2 as

$$d_{v_1 \to v_2} = (d_1 K^{-1}, d_2, R),$$

  where $k \in_R \{0,1\}^l, K = E_2(k) \in \mathbb{G}$ and $R \leftarrow \mathsf{Encrypt'}(\mu, v_2, k)$. We define $\mathsf{Encrypt'}$ the same as **Encrypt** except that it appends '1' to the message.
- **Reencrypt**$(\mu, d_{v_1 \to v_2}, C_{v_1})$**:** Let $d_{v_1 \to v_2} = (\hat{d}_1, d_2, R)$ and $C_{v_1} = (c_1, c_2, c_3, c_4, vk, \sigma)$. Check if $\mathsf{Vrfy}_{vk}((c_1, c_2, c_3, c_4), \sigma) \stackrel{?}{=} 1$. If not, output $\perp$. Otherwise, compute $d'_1 = \hat{d}_1 F_2(vk)^{r'}, d'_2 = g^{r'}$ and

$$C_{v_2} = (C_{v_1}, R, d'_1, d_2, d'_2)$$

  where $r' \in_R \mathbb{Z}_p$.

– **Decrypt**(μ, d_v, C_v)**:**
If C_v is a regular encryption, let $d_v = (d_1, d_2)$ and $C_v = (c_1, c_2, c_3, c_4, vk, \sigma)$. Check if $\mathsf{Vrfy}_{vk}((c_1, c_2, c_3, c_4), \sigma) \stackrel{?}{=} 1$. If not, output $\perp$. Otherwise, compute $d_1' = d_1 F_2(vk)^{r'}, d_2' = g^{r'}$ where $r' \in_R \mathbb{Z}_p$ and decrypt C_v:

$$M = c_1 \frac{e(d_2, c_3)e(d_2', c_4)}{e(d_1', c_2)}.$$

If C_v is a re-encrypted ciphertext, let $C_v = (C_1, R, d_1', d_2, d_2')$ and $C_1 = (c_1, c_2, c_3, c_4, vk, \sigma)$. Check if

- $\mathsf{Vrfy}_{vk}((c_1, c_2, c_3, c_4), \sigma) \stackrel{?}{=} 1$, and
- $e(d_1' k, g) \stackrel{?}{=} e(g_1, g_2)e(F_1(v_1), d_2)e(F_2(vk), d_2')$.

If not, output $\perp$. Otherwise, decrypt C_v by performing

$$k \leftarrow \mathsf{Decrypt'}(\mu, d_v, R), \quad K = E_2(k), \quad \text{and} \quad M = c_1 \frac{e(d_2, c_3)e(d_2', c_4)}{e(d_1' K, c_2)}.$$

Let $m||b = E_1^{-1}(M)$. If $b = 0$, output m; otherwise, output $\perp$. We define $\mathsf{Decrypt'}$ the same as **Decrypt** except that it outputs m if the decrypted message ends with 1 and outputs $\perp$ if it ends with 0.

We append an extra bit to the message in order to distinguish between the encryption of messages and the encryption of re-encryption key. Then the adversary can't treat the challenge ciphertext as R to the decryption oracle and get some information.

As the argument of IB-PRE-I, the proxy only needs to re-encrypt R to convert the re-encrypted ciphertext to others. If the recipient can decrypt the ciphertext to get k, then he can get m. This meets the multi-use property.

5.2 Security

The security proof of IB-PRE-II is like the proof of IB-PRE-I, except that the adversary $\mathcal{A}$ can make additional queries to REENCRYPT and DECRYPT oracles.

Theorem 4. *Suppose there is an adversary $\mathcal{A}$ that has advantage ϵ against the game $Exp^{\mathcal{A},\mathsf{IND\text{-}PrID\text{-}CCA}}$. Then there is an algorithm $\mathcal{B}$ that breaks Wa-CCA-IBE with advantage at least*

$$Adv_{\mathcal{B}}^{\mathsf{IND\text{-}ID\text{-}CCA}} \geq \epsilon / e(1 + q_E),$$

where q_E is the maximal number of $\mathcal{A}$'s queries to EXTRACT*, and e is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.*

Proof. Assume that there is an adversary $\mathcal{A}$ breaking IB-PRE-II scheme. We construct an algorithm $\mathcal{B}$ to break Wa-CCA-IBE scheme. Given the public parameter μ of Wa-CCA-IBE scheme, $\mathcal{B}$ performs the following steps. Note that $\mathcal{B}$ maintains a table with tuples (β, v_1, v_2, d), where d is the re-encryption key. Let $*$ denote the wildcard. Without loss of generality, we assume that an input is queried to an oracle only once.

1. Setup. Give the parameter μ to $\mathcal{A}$.
2. Query phase 1. $\mathcal{A}$ can make the following queries.
   (a) EXTRACT(v): $\mathcal{B}$ first generates a random coin β so that $\Pr[\beta = 1] = \delta$ for some δ that will be determined later. If $\beta = 0$, or $(0, v, *, *)$ or $(0, *, v, *)$ already exists on the table, $\mathcal{B}$ outputs a random bit and aborts. Otherwise, $\mathcal{B}$ sends the query to the key extraction oracle of Wa-CCA-IBE to get d_v, and returns d_v to $\mathcal{A}$. $\mathcal{B}$ also records $(1, v, v, \perp)$ on the table.
   (b) RKEXTRACT(v_1, v_2): $\mathcal{B}$ chooses a random coin β as in the EXTRACT. If $\beta = 1$, or $(1, v_1, v_1, \perp)$ or $(1, v_2, v_2, \perp)$ already exists on the table, $\mathcal{B}$ queries the key extraction oracle of Wa-CCA-IBE for v_1, and then computes the re-encryption key as the original scheme and returns it to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns a random re-encryption key $d_{v_1 \to v_2} = (x, y, \mathsf{Encrypt}'(\mu, v_2, z))$ for $x, y \in_R \mathbb{G}$ and $z \in_R \{0,1\}^l$. Finally, $\mathcal{B}$ records the tuple $(\beta, v_1, v_2, d_{v_1 \to v_2})$ on the table.
   (c) REENCRYPT(v_1, v_2, C_{v_1}): Let $C_{v_1} = (c_1, c_2, c_3, c_4, vk, \sigma)$. If $(*, v_1, v_2, *)$ does not exist on the table, $\mathcal{B}$ performs RKEXTRACT(v_1, v_2) to get the re-encryption key $d_{v_1 \to v_2}$. Then for the tuple $(*, v_1, v_2, d_{v_1 \to v_2})$, $\mathcal{B}$ uses $d_{v_1 \to v_2}$ to re-encrypt the ciphertext as the real scheme.
   (d) DECRYPT(v, C_v): If C_v is a regular encryption, $\mathcal{B}$ sends the ciphertext to the decryption oracle of Wa-CCA-IBE for v, and returns the plaintext to $\mathcal{A}$ if and only if it ends with '0'. If C_v is a re-encrypted ciphertext, let $C_v = (C_{v_1}, R, d_1', d_2, d_2')$ and $C_{v_1} = (c_1, c_2, c_3, c_4, vk, \sigma)$. If $(0, v_1, v, \tilde{d})$ exists on the table for any v_1 and $\tilde{d} = (\tilde{d}_1, \tilde{d}_2, \tilde{d}_3)$, $\mathcal{B}$ checks whether

$$e(d_1', g) \stackrel{?}{=} e(\tilde{d}_1, g)e(F_2(vk), d_2'), \quad d_2 \stackrel{?}{=} \tilde{d}_2 \quad \text{and} \quad R \stackrel{?}{=} \tilde{d}_3.$$

   If the equations hold, $\mathcal{B}$ sends C_{v_1} to the decryption oracle of Wa-CCA-IBE for v_1, and returns the plaintext to $\mathcal{A}$ if and only if it ends with '0'. Otherwise, $\mathcal{B}$ sends R to the decryption oracle of Wa-CCA-IBE scheme for v to get $k||1$ (otherwise, return $\perp$) and computes

$$M = c_1 \frac{e(d_2, c_3)e(d_2', c_4)}{e(d_1' K, c_2)}$$

   for $K = E_2(k)$. If $E_1(M) = m||0$ for some m, $\mathcal{B}$ returns m to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns $\perp$.
3. Challenge. $\mathcal{A}$ sends (v^*, m_0, m_1) to $\mathcal{B}$, if $(1, v^*, v', k)$ exists on the table for any v' and k, $\mathcal{B}$ randomly outputs a bit and aborts. Otherwise, $\mathcal{B}$ sends the challenge $(v^*, m_0||0, m_1||0)$ to the challenger of Wa-CCA-IBE. When the challenger returns ciphertext C^*, $\mathcal{B}$ returns C^* to $\mathcal{A}$.
4. Query phase 2. $\mathcal{A}$ continues making the following queries, except for the restrictions described in Definition 2.
   (a) EXTRACT(v): $\mathcal{B}$ answers queries as in the Query phase 1.
   (b) RKEXTRACT(v_1, v_2): For all $v_1 \neq v^*$, $\mathcal{B}$ queries the key extraction oracle of Wa-CCA-IBE for v_1, and then computes the re-encryption key as the original scheme and returns it to $\mathcal{A}$. For $v_1 = v^*$, $\mathcal{B}$ returns a random

re-encryption key $d_{v_1 \to v_2} = (x, y, \mathsf{Encrypt}'(\mu, v_2, z))$ for $x, y \in_R \mathbb{G}, z \in_R \{0,1\}^l$, and records the tuple $(0, v, v', d_{v_1 \to v_2})$ on the table.

(c) $\textsc{Reencrypt}(v_1, v_2, C_{v_1})$: $\mathcal{B}$ answers queries as in the Query phase 1.

(d) $\textsc{Decrypt}(v, C_v)$: $\mathcal{B}$ answers queries as in the Query phase 1.

5. Guess. When $\mathcal{A}$ outputs the guess b', $\mathcal{B}$ outputs b'.

Since $\mathcal{B}$ aborts the game in the same conditions, by the proofs of Theorem 3 and Lemma 1, $\mathcal{B}$'s advantage is at least $\epsilon/e(1 + q_E)$. □

6 Conclusion

In this paper, we propose two identity-based proxy re-encryption schemes without random oracles. The schemes both satisfy the properties of unidirectionality, non-interactivity and multi-use. The security of our schemes are based on the underlying IBE schemes. The solutions answer the open problems left in the previous work.

Acknowledgement

We first thank anonymous reviewers for giving us many useful suggestions. Also, we are grateful to Sherman S.M. Chow for pointing out some security flaws in our manuscript and giving a help to fix the problem.

References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Proceedings of the Network and Distributed System Security Symposium (NDSS '05) The Internet Society (2005)
2. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Katz, J.: Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM 51(4), 557–594 (2004)

8. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
9. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. Cryptology ePrint Archive, Report 2007/171 (2007)
10. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Proceedings of Applied Cryptography and Network Security 2007 (ACNS '07). LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
11. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007)
12. Ivan, A.-A., Dodis, Y.: Proxy cryptography revisited. In: Proceedings of the Network and Distributed System Security Symposium (NDSS '03) The Internet Society (2003)
13. Jakobsson, M.: On quorum controlled asymmetric proxy re-encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 112–121. Springer, Heidelberg (1999)
14. Mambo, M., Okamoto, E.: Proxy cryptosystems: delegation of the power to decrypt ciphertexts. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E80-A(1), 54–63 (1997)
15. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Strongly-Secure Identity-Based Key Agreement and Anonymous Extension*

Sherman S.M. Chow[1] and Kim-Kwang Raymond Choo[2,**]

[1] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[2] Australian Institute of Criminology
GPO Box 2944, Canberra ACT 2601, Australia
raymond.choo@aic.gov.au

Abstract. We study the provable security of identity-based (ID-based) key agreement protocols. Although several published protocols have been proven secure in the random oracle model, only a weak adversarial model is considered – the adversary is not allowed to ask Session-Key Reveal queries that will allow the adversary to learn previously established session keys. Recent research efforts devoted to providing a stronger level of security require strong assumptions, such as assuming that the simulator has access to a non-existential computational or decisional oracle. In this work, we propose an ID-based key agreement protocol and prove its security in the widely accepted indistinguishability-based model of Canetti and Krawczyk. In our proof, the simulator does not require access to any non-existential computational or decisional oracle. We then extend our basic protocol to support ad-hoc anonymous key agreement with bilateral privacy. To the best of our knowledge, this is the first protocol of its kind as previously published protocols are for fixed group and provide only unilateral privacy (i.e., only one of the protocol participants enjoy anonymity).

Keywords: Key agreement, identity-based cryptography, reveal query, provable security, anonymity.

1 Introduction

To establish secure communications over an insecure communication channel, a common secret (session) key to be shared among the communicating parties is often established using key establishment protocols. Key establishment protocols

* The full version is available at IACR Cryptology ePrint Archive [19].

** The views and opinions expressed in this paper are those of the author and do not reflect those of the Australian Government or the Australian Institute of Criminology. This research was not undertaken as part of the author's work at the Australian Institute of Criminology.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 203–220, 2007.

can be broadly categorised into key agreement protocols or key transport protocols depending on the nature of the session key (whether input to the session key is required from only one party or all the participating parties).

The basis of many key establishment protocols relies on the Diffie–Hellman key exchange and the RSA algorithm (e.g. see [14, Chapter 2]). In recent years, elliptic curve cryptography has emerged as a promising branch of public-key cryptography particularly due to its potential for offering similar security to established public-key cryptosystems at reduced key sizes. We also observe an emerging trend in the use of identity-based cryptography, such as a large number of identity-based key agreement protocols based on pairings [5]. The public keys in ID-based system are arbitrary bit-strings and can include any descriptive information such as temporal information. The corresponding private key is then generated by a trusted key generation center (KGC). The strength of ID-based systems in terms of a simplified key management system (i.e., no public key certificates required) is also one of its weaknesses. Users are not allowed to generate their own private keys and therefore key escrow is inevitable. Key agreement protocols help to establish a session key that may not be under KGC's escrow.

We now highlight two other on-going research problems in the design of ID-based key agreement protocols, the focus of this paper.

Security issues: Session-Key Reveal ***and*** Session-State Reveal ***queries***
The purported security of many ID-based protocols for two parties is proven in a weak variant of Bellare–Rogaway (BR) model [3] in which the adversary is not allowed to ask any Session-Key Reveal[1] query [5]. Protocols proven secure in such a restricted model (hereafter referred to as the wBR model) do not provide the known-key security attribute, meaning that compromise of previously accepted session keys may affect the security of a non-related session.

To better explain the Session-Key Reveal (and the stronger notion of Session-State Reveal queries in the Canetti–Krawczyk model), we recall the unauthenticated Diffie-Hellman key exchange protocol as described in Figure 1. In the protocol, all arithmetic is performed modulo a large prime p with q being the prime order of g, $\in_R$ denotes choosing an element uniformly at random from the corresponding domain, $\mathcal{K}$ denotes a key derivation function (which can be realized by a hash function mapping to the secret key domain of some symmetric cryptographic scheme), and sk denotes the session key established at the conclusion of the protocol execution.

We now execute the protocol described in Figure 1 twice. Two independent sessions with the respective session keys, $sk_1 = g^{ab}$ and $sk_2 = g^{a'b'}$, where $a \neq a', b'$ and $b \neq a', b'$, were established. We assume that there exists a malicious adversary who is interested to learn the session key associated with one of the sessions, e.g., $sk_1 = \mathcal{K}(g^{ab})$ – the session key associated with the first session.

[1] The Session-Key Reveal query allows the adversary to learn previously established session keys.

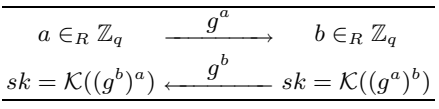


Fig. 1. Diffie–Hellman Protocol

- In a security model that allows the adversary to ask the Session-Key Reveal query, the adversary is allowed to learn session key associated with any non-related session, i.e., $sk_2 = \mathcal{K}(g^{a'b'})$.
- In a security model that allows the adversary to ask the Session-State Reveal query, the adversary is allowed to learn the ephemeral parameters of any non-related sessions. In this case, the adversary is allowed to learn either the ephemeral DH keys, a' and b', or the keying material $g^{a'b'}$, if they have not been erased from the internal state of the respective entity.

The significance of Session-State Reveal queries stems from the fact that a user may decide to store the pre-computed results to be used in future session key establishment for efficiency. These parameters, often not protected as securely as the long-term private key, may be exploited by the adversary. Such a query is designed to consider the leakage of such ephemeral parameters.

It is common practice to prove the strongest security that we can claim about any cryptographic scheme and this seems a sound principle to follow in the case of ID-based key agreement protocols. It is therefore not surprising that we advocate the importance of proving ID-based protocols secure in a security model that allows the adversary to ask both the Session-Key Reveal and Session-State Reveal queries. Protocols proven secure in such a model will also assure protocol implementers that they provide known-key security attribute and provide resilience against the leakage of ephemeral parameters.

Privacy issues: Confidentiality of identity

Anonymity is required in many applications to ensure that the identifying information about the user is not revealed. This concept is also useful and applicable to key agreement protocol. Suppose two entities, $\mathcal{U}$ and $\mathcal{V}$, want to exchange confidential messages. In anonymous key agreement protocols such as the protocols of Boyd and Park [7] and of Shoup [29], $\mathcal{U}$'s identity is not known to anyone in the network except $\mathcal{V}$ – the recipient entity in the key agreement protocol.

This work considers anonymity from a slightly different perspective. Although $\mathcal{V}$ knows that $\mathcal{U}$ is a member of a group of users, $\mathcal{V}$ is unable to confirm the actual identity of $\mathcal{U}$. This class of protocol is useful when $\mathcal{V}$ only needs to ensure the membership of the sender, but not the identity of the user perhaps due to privacy issues. Our protocol provides deniability [6] for any user who has taken part in a protocol run to deny that this was the case, since any one can simulate runs of the protocol involving any other potential user.

2 Related Work and Our Contributions

2.1 Session-Key Reveal and Session-State Reveal Queries

Recent research efforts have been devoted towards designing protocols that can be proven secure in a model that allows the Session-Key Reveal queries. For example, the ID-based protocols of Chen and Kudla [9] and McCullagh and Barreto [25] were improved [17] to ensure that these protocols can be proven secure in a less restrictive sense (the adversary is allowed to ask Session-Key Reveal queries in most cases) in the random oracle model, assuming bilinear Diffie-Hellman problem is intractable. The technicality of not being able to answer reveal queries in some special sessions can be resolved using the gap assumption – the underlying computational problem is intractable even with the help of a corresponding decisional oracle.

Using the gap assumption, Kudla and Paterson [23] propose a generic transformation turning two-party Diffie–Hellman-based protocols proven secure in the wBR model to one in the full BR model. This is also applicable to two-party ID-based protocols such as the protocols of Chen and Kudla [9] and McCullagh and Barreto [25]. However, gap assumption in [9] and [25] means the simulator has access to a decisional *bilinear* Diffie-Hellman oracle (in contrast with decisional Diffie-Hellman oracle that can be realized by some classes of pairing). This result also matches with the observation raised by Chow ([18] as cited in [17]).

Along somewhat similar line, Wang [32] proposes a protocol based on a decisional problem by using a *computational oracle* to support the Session-Key Reveal queries. Again, the simulation in this proof requires the existence of a special oracle. Finally, we note that Cheng *et al.* [12] introduce the concept of coin queries that forces the adversary to reveal its ephemeral secret, and thus making Session-Key Reveal possible. Their approach is restricted in the sense that the possibility of breaking a protocol without knowing the ephemeral secret (which is possible in a real world attack) is not modelled.

The Session-State Reveal query in the Canetti–Krawczyk model (hereafter referred to as the CK model) [8] allows an adversary to learn the ephemeral parameters associated with a particular session. An example of a protocol secure in this stronger model[2] is the HMQV protocol [22], which is the "hashed" variant of the MQV protocol[3]. The basic version of HMQV is proven secure even if the adversary is allowed to ask Session-Key Reveal queries under the computational Diffie-Hellman assumption. The enhanced version of HMQV is proven secure even when the adversary learns the ephemeral Diffie–Hellman key associated with any non-target sessions, under the gap Diffie-Hellman assumption and knowledge of exponent assumption [2]. No security claim is, however, made about the availability of the keying material for the derivation of the session key.

Our contribution: High-performance ID-based key agreement protocol
We propose a new ID-based key agreement protocol. Security assurance of the protocol is provided in the stronger CK model, which allows the adversary to

[2] The relative strengths between the BR and CK models are discussed in [16].
[3] MQV's security is analyzed [24], without consideration of Session-State Reveal query.

ask Session-Key Reveal queries in *all* cases, and Session-State Reveal queries in most cases, *without* employing any gap assumption. We show how to provide *KGC forward secrecy* by making minor modifications to the (basic) protocol. Additional parameters included in our session state definition are the ephemeral Diffie–Hellman (DH) key of the outgoing DH values and the keying material for the key derivation. Among the ID-based two-party protocols surveyed in [5], our proposed protocols achieve the strongest security properties without compromising on efficiency.

2.2 Anonymous Key Agreement Protocols

To illustrate the usefulness of our proposed key agreement protocols, we now consider the scenario of delegates making and receiving phone calls on their mobile phones while international roaming. Before secure roaming can be established, the service provider must verify whether the roaming user is a legitimate subscriber with the respective home server. Conventional anonymous roaming mechanisms [1,26] are rather inefficient as users would have to wait online while foreign telecommunication network communicates with the original home server to authenticate the users. These geographically distributed servers also generate extra network traffic during this process. At the same time, it is inconvenient to constantly *renew* the alias in an *unlinkable* manner to hide the identities.

Our proposed key agreement protocols with the anonymity feature allow user to "hide" among a group of subscribers associated with the same home server. Moreover, after the home server has issued sets of matching public/private key pair at the very beginning, the home server is no longer required to be online. Our approach does not, however, appear to be *scalable* if one needs to hide among all (a potentially large set of) legitimate subscribers, and may not be *flexible* since it is natural that the set of subscribers is constantly changing. Both issues can be readily solved without an *a priori* group formation step. For example, any legitimate user will be able to spontaneously conscript an arbitrary group of users (i.e., without cooperation from other parties in the group) for each session. Such *ad-hoc* group formation empowers a user to have full control over the level of anonymity desired during the secure roaming establishment process.

Although alias should also be used in our approach so that the list of users can be made available to users without revealing any user information, no renewal of alias (and possibly renewal of credential) is necessary as different invocations are unlinkable (guaranteed by the unconditional anonymity of our protocol).

Our contribution: Key agreement protocol with bilateral privacy
Motivated by the various applications of anonymous roaming and our observation that existing research (e.g., see [11]) appears to focus only on *unilateral* identity privacy (i.e., only one protocol participant enjoys anonymity), we propose a secure key exchange among anonymous users in different spontaneous groups. Spontaneity and bilateral privacy features in our proposed protocol are particularly applicable in ad-hoc group communication settings. Furthermore, as noted in the literature of ID-based ring signature (e.g., [20]), ID-based solution

provides a higher level of spontaneity and efficiency than conventional public key cryptosystem since one can conscript virtually anyone and no verification of public key certificates is required. With these benefits in mind, we introduce the notion of ID-based ad-hoc anonymous key agreement with *bilateral* privacy, which is realized by an extension of our basic protocol. Note that our approach is fundamentally different from that of Cheng *et al.* [11].

3 Number Theoretic Assumptions

Let $\mathbb{G}$ be an additive group of prime order q and $\mathbb{G}_T$ be a multiplicative group also of order q. We assume the existence of an efficiently computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that

1. There is an known element $P \in \mathbb{G}$ satisfying $\hat{e}(P, P) \neq 1_{\mathbb{G}_T}$.
2. For $Q, W, Z \in \mathbb{G}$, both $\hat{e}(Q, W + Z) = \hat{e}(Q, W) \cdot \hat{e}(Q, Z)$ and $\hat{e}(Q + W, Z) = \hat{e}(Q, Z) \cdot \hat{e}(W, Z)$.

Definition 1 (Interactive Game with a BDH Challenger [11]). *Let $\mathcal{A}$ be a pair of probabilistic polynomial-time (PPT) algorithms $(\mathcal{A}_1(r_1; \ldots), \mathcal{A}_2(r_2; \ldots))$, where r_i is used by $\mathcal{A}_i$ as the random tape, that engages with a challenger in the following game. Let (P, aP, bP, cP) be the BDH instance where $P, aP, bP, cP \in \mathbb{G}$ and $a, b, c \in \mathbb{Z}_q^*$. The game is defined as follows.*

Stage 1: $(X, \sigma) \leftarrow \mathcal{A}_1(r_1; P, aP, bP, cP, \hat{e}, \mathbb{G}, \mathbb{G}_T, q)$ *(σ denotes some state)*
Interactive Part: *After seeing X, challenger returns a random $h \leftarrow_R \mathbb{Z}_q^*$.*
Stage 2: $K \leftarrow \mathcal{A}_2(r_2; h, \sigma)$.

We say that the adversary, $\mathcal{A}$, wins the game if it computes $K = \hat{e}(aP, X + hbP)^c$.

If X is determined *after* seeing h, the problem is easy since one can set $X = rP - hbP$ for $r \in_R \mathbb{Z}_q^*$, and returns $K = \hat{e}(aP, cP)^r$. It explains the game's interactive nature.

The following lemma says that if the BDH problem is hard, any adversary can only have a negligible advantage in winning the interactive BDH game. The proof is similar to the one presented in [11].

Lemma 1 (Interactive BDH Game Assumption). *For any adversary with PPT algorithm $(\mathcal{A}_1, \mathcal{A}_2)$ with advantage $\epsilon(k)$ to win the interactive BDH game, there exists an algorithm that solves BDH problem with probability $\epsilon(k)^2$.*

Proof. Given a BDH problem instance $(P, aP, bP, cP, \hat{e}, \mathbb{G}, \mathbb{G}_T, q)$, we construct a BDH solver $\mathcal{B}$ making use of $(\mathcal{A}_1, \mathcal{A}_2)$ as follows.

$\mathcal{B}$ starts by choosing two elements h and h' randomly from $\mathbb{Z}_q^*$. $\mathcal{B}$ calls $(X, \sigma) \leftarrow \mathcal{A}_1(r_1; P, aP, bP, cP, \hat{e}, \mathbb{G}, \mathbb{G}_T, q)$ and $K \leftarrow \mathcal{A}_2(r_2; h, \sigma)$. $\mathcal{B}$ now rewinds the adversary backs to the point before $\mathcal{A}_2$ is called. $\mathcal{A}_2$ is then executed again with h' to get $K \leftarrow \mathcal{A}_2(r_2; h', \sigma)$. Since h and h' are chosen independently from X and σ, the probability each of two executions of $\mathcal{A}_2$ returns a valid answer is at least $\epsilon(k)$. Under such condition, $K = \hat{e}(aP, X + hbP)^c$ and $K' = \hat{e}(aP, X + h'bP)^c$. Ignoring the negligible probability that $h = h'$, $\hat{e}(aP, bP)^c$ can be obtained by $(K/K')^{(h-h')}$, i.e., $\mathcal{B}$ solves BDH problem with probability $\epsilon(k)^2$. □

In the proofs of Kudla and Paterson [23] and Wang [32], the required (decisional BDH) oracle, in which the simulator has access, has no known polynomial time realization. Their assumptions are non-falsifiable whilst in our case, we only assume the BDH problem is intractable, something that can be falsified.

We also consider a variant of the BDH problem, the Modified Bilinear Diffie-Hellman(MBDH) problem, for the proof of our escrow-free protocol.

Definition 2 (Modified (Computational) Bilinear Diffie-Hellman (MBDH) Problem [21]). *Given* $(P, aP, bP, cP, c^{-1}P)$, *output* $\hat{e}(P,P)^{abc} \in G_2$.

Computational and decisional MBDH problems were first proposed in [21] to realize the first ID-based signcryption scheme with forward-secrecy and public ciphertext authenticity. In this paper, we reduce the security of our escrow-free protocol to an interactive MBDH assumption, which is defined in a way similar to Definition 1 (adding an extra element to be supplied to the adversary) to support KGC forward-secrecy. We have the following result as described by Lemma 2.

Lemma 2 (Interactive MBDH Game Assumption). *For any adversary with PPT algorithm* $(\mathcal{A}_1, \mathcal{A}_2)$ *with advantage* $\epsilon(k)$ *to win the interactive MBDH game, there exists an algorithm that solves MBDH problem with probability* $\epsilon(k)^2$.

4 High Performance ID-Based Key Agreement Protocol

4.1 Basic Construction

Setup: On input a security parameter k, KGC uses a BDH instance generator to generate $(\mathbb{G}, \mathbb{G}_T, \hat{e})$ where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order q and $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is the pairing function. KGC also chooses two cryptographic hash functions $\mathcal{H} : \{0,1\}^n \rightarrow \mathbb{G}$ and $\mathcal{H}_0 : \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and a key derivation function $\mathcal{K}$. All three of these are modelled as random oracles. Then KGC randomly chooses an arbitrary generator P of $\mathbb{G}$. An element s is randomly chosen from $\mathbb{Z}_q^*$ as the KGC's master secret, and the corresponding public key is $P_{pub} = sP$. Finally, the set of public parameters is published as $\mathsf{params} = \langle \mathbb{G}, \mathbb{G}_T, q, \hat{e}, P, P_{pub}, \mathcal{H}, \mathcal{H}_0, \mathcal{K} \rangle$.

Extract ([4]): On inputs an identity ID_A and a master secret s, the public key Q_A (for A) is set as $\mathcal{H}(\mathsf{ID}_A)$, and the corresponding private key S_A is $s\mathcal{H}(\mathsf{ID}_A)$.

Key Agreement: Our proposed high performance identity-based key agreement protocol is described in Figure 2. The notation used in the protocol is as follows: (Q_U, S_U) denotes the public/private key pair for protocol participant U, sk_U and sid_U denote the session key and session identifier for protocol participant U respectively and $||$ denotes the concatenation of messages.

4.2 Security Evaluation: An Overview

The simulator, $\mathcal{S}$, knows how to answer all but one Corrupt queries, ID_J. The hard problem will be embedded in one of the sessions having ID_J as the responder.

Initiator A		Responder B
$a \in_R \mathbb{Z}_q^*; W_A := aQ_A$	$\xrightarrow{ID_A, W_A}$	$b \in_R \mathbb{Z}_q^*; W_B := bQ_B$
$h_A := \mathcal{H}_0(W_A, \mathsf{ID}_B)$		$h_A := \mathcal{H}_0(W_A, \mathsf{ID}_B)$
		$sid_B := ID_A \| W_A \| ID_B \| W_B$
		$sk_B := \mathcal{K}(\hat{e}(W_A + h_A Q_A, (b + h_B) S_B))$
$h_B := \mathcal{H}_0(W_B, \mathsf{ID}_A)$	$\xleftarrow{ID_B, W_B}$	$h_B := \mathcal{H}_0(W_B, \mathsf{ID}_A)$
$sid_A := ID_A \| W_A \| ID_B \| W_B$		
$sk_A := \mathcal{K}(\hat{e}((a + h_A) S_A, W_B + h_B Q_B))$		
$sk_A = \mathcal{K}(\hat{e}(Q_A, Q_B)^{s(a+h_A)(b+h_B)}) = sk_B$		

Fig. 2. Proposed high-performance identity-based key agreement protocol

Note that neither the Session-Key Reveal queries nor the Session-State Reveal queries are allowed for this test session. For all other sessions having ID_J as the responder, $\mathcal{S}$ can correctly answer the queries asked since all state information and the private key of the initiator ID_I are known to $\mathcal{S}$.

The tricky part is answering queries directed at the sessions where ID_J acts as the initiator. $\mathcal{S}$ can, however, faithfully simulate the protocol execution by defining W_J before the output of the corresponding random oracle query $\mathcal{H}_0(W_J, \mathsf{ID}_K)$ is defined. $\mathcal{S}$ can then compute the session key in some way different from the protocol specification to answer the Session-Key Reveal query. As an abnormal way is used, answering the Session-State Reveal query correctly is not possible and this is our only restriction on simulating the Session-State Reveal queries.

Theorem 1. *The protocol described in Figure 2 is secure assuming that the BDH problem is hard*[4] *and $\mathcal{H}$, $\mathcal{H}_0$, and $\mathcal{K}$ are modelled as random oracles.*

Proof. Assuming that there exists an adversary $\mathcal{A}$ with a non-negligible advantage against our protocol described in Figure 2, we construct a simulator, $\mathcal{S}$, against the interactive game with a BDH challenger (the BDH problem instance is (P, xP, yP, zP) and the last part of the challenge is h), using $\mathcal{A}$ as a subroutine. $\mathcal{S}$ now simulates the view of $\mathcal{A}$ by answering the following queries of $\mathcal{A}$.

Setup: xP is assigned to be the public key of the KGC.

$\mathcal{H}$ queries: If an $\mathcal{H}$ query is previously asked, then the stored answer in the list $L_{\mathcal{H}}$ will be returned. Denote the I^{th} distinct $\mathcal{H}$ query by ID_I. For ID_J, $\mathcal{S}$ responses with yP; otherwise, $\mathcal{S}$ chooses $r_i \in_R \mathbb{Z}_q^*$, stores it in the list $L_{\mathcal{H}}$ along with ID_I, and outputs $r_i P$.

$\mathcal{H}_0$ queries: $\mathcal{S}$ maintains a list $L_{\mathcal{H}_0}$ to ensure that previously asked queries would receive the same answer. However, special value may be plugged into the list in

[4] Recall that in the result of Lemma 1, we assume the BDH problem is hard. By doing so, we also assume a negligible advantage in the interactive BDH game.

the simulation of the Send queries with ID_J as the initiator and ID_K as the responder.

$\mathcal{K}$ *queries*: $\mathcal{S}$ just needs to ensure the random oracle property of $\mathcal{K}$, by maintaining a list $L_{\mathcal{K}}$ to ensure that previously asked queries will receive the same answer. It can be seen from the rest of the proof that the simulator knows the keying materials for all sessions, while the test session is the only exception.

Corrupt *queries*: The simulation fails (event I) if the request is ID_J, otherwise the corresponding r_i is retrieved from the list $L_{\mathcal{H}}$ and $r_i(xP)$ is returned.

Send *queries (*ID_I *as initiator and* ID_J *as responder)*: Since $\mathcal{S}$ can compute the private key of ID_I so the simulation can be done as a typical protocol invocation. Except for the following special handling for the N^{th} invocation, τ is chosen randomly from $\mathbb{Z}_q^*$ and $W_{I,N} = r_I\tau(zP)$ is returned. After $W_{J,N}$ is obtained, if $(W_{J,N}, \mathsf{ID}_I)$ can be found in list $L_{\mathcal{H}_0}$, the simulation fails (event II). Otherwise, $\mathcal{S}$ dumps all maintained lists and system parameters to the tape σ, then outputs (X, σ) where $X = W_{J,N}$. The interactive BDH challenger returns $h \in_R \mathbb{Z}_q^*$. $\mathcal{S}$ reconstructs all the lists and system parameters from σ, and set $\mathcal{H}_0(W_{J,N}, \mathsf{ID}_I) = h$, which is also denoted as $h_{J,N}$.

Send *queries (*ID_J *as initiator and* ID_K *as responder)*: In this case, $\mathcal{S}$ knows neither the private key of the initiator ID_J, nor the ephemeral Diffie-Hellman key of the responder ID_K. However, $\mathcal{S}$ can still do a faithful simulation by manipulating the random oracle. Suppose it is the ℓ^{th} invocation of the protocol initiated by ID_J and responded with ID_K. $\mathcal{S}$ selects $\alpha_\ell, h_{J,\ell} \in_R \mathbb{Z}_q^*$, responses with $W_{J,\ell} = \alpha_\ell P - h_{J,\ell}Q_J$, and stores h_J as the response of $\mathcal{H}_0$ corresponding to the query $(W_{J,\ell}, \mathsf{ID}_K)$. α_ℓ is also stored in the auxiliary list corresponding to the $\Pi^n_{J,K}$ session.

Session-Key Reveal *queries*: For session having ID_I as initiator and ID_J as responder, and if this is not the N^{th} invocation, $\mathcal{S}$ simply uses the private key of ID_I to answer the query asked by $\mathcal{A}$ since $\mathcal{S}$ knows the ephemeral Diffie-Hellman key chosen; otherwise, it fails (event III). For the case $(\mathsf{ID}_J, \mathsf{ID}_K)$, suppose $h_{J,\ell} = \mathcal{H}_0(W_{J,\ell}, \mathsf{ID}_K)$ and $h_{K,\ell} = \mathcal{H}_0(W_{K,\ell}, \mathsf{ID}_J)$. $\mathcal{S}$ retrieves α_ℓ and returns $\mathcal{K}(\hat{e}(\alpha_\ell(xP), W_{K,\ell} + h_{K,\ell}Q_K))$. Consistency can be easily seen:

$$\begin{aligned}\mathcal{K}(\hat{e}(\alpha_\ell(xP), W_{K,\ell} + h_{K,\ell}Q_K)) &= \mathcal{K}(\hat{e}(\alpha_\ell P, W_{K,\ell} + h_{K,\ell}Q_K)^x) \\ &= \mathcal{K}(\hat{e}(\alpha_\ell P - h_{J,\ell}Q_J + h_{J,\ell}Q_J, W_{K,\ell} + h_{K,\ell}Q_K)^x) \\ &= \mathcal{K}(\hat{e}(W_{J,\ell} + h_{J,\ell}Q_J, W_{K,\ell} + h_{K,\ell}Q_K)^x).\end{aligned}$$

Session-State Reveal *queries*: For session having ID_I as initiator and ID_J as responder, it is trivial to obtain the ephemeral Diffie-Hellman key, except for the N^{th} invocation where $\mathcal{S}$ will fail (event IV). For $(\mathsf{ID}_J, \mathsf{ID}_K)$, it is not supported.

$\mathcal{S}$ knows all the outgoing and incoming DH values, even for the N^{th} invocation between ID_I and ID_J and invocations between ID_J and arbitrary ID_K. $\mathcal{S}$ also knows the keying material for all sessions, except the N^{th} invocation (event IV).

Test *queries*: Suppose $h_{I,N} = \mathcal{H}_0(W_{I,N}, \mathsf{ID}_J)$ and $h_{J,N} = \mathcal{H}_0(W_{J,N}, \mathsf{ID}_I) = h$. If $\mathcal{A}$ does not choose the session $\Pi^N_{I,J}$, $\mathcal{S}$ aborts (event V). $\Pi^N_{I,J}$ should hold a session key of the following form.

$$\mathcal{K}(\hat{e}(W_{I,N} + h_{I,N}Q_I, W_{J,N} + h_{J,N}Q_J)^x) = \mathcal{K}(\hat{e}(r_I\alpha(zP) + h_{I,N}r_IP, X + hyP)^x)$$
$$= \mathcal{K}(\hat{e}((\alpha z + h_{I,N})r_IP, X + hyP)^x) = \mathcal{K}(\hat{e}(xP, X + hyP)^{(\alpha z + h_{I,N})r_I}).$$

$\mathcal{S}$ cannot compute $\mathcal{K}(\hat{e}(xP, X + h(yP))^{z(r_I\alpha)})$ by itself without the assistance of $\mathcal{A}$. Therefore, $\mathcal{S}$ is unable to return the real session key. A random key drawn from session key distribution (range of $\mathcal{K}$) will be returned instead.

Answering interactive BDH challenger: If $\mathcal{S}$ does not abort and $\mathcal{A}$ is able to distinguish between real session key and random session key (with probability $\epsilon(k)$), then $\mathcal{A}$ must have queried the key derivation oracle $\mathcal{K}$ for the keying material $\hat{e}(xP, X + hyP)^{(\alpha z + h_{I,N})r_I} = \hat{e}(xP, X + h(yP))^{z(r_I\alpha)}\hat{e}(xP, X + h(yP))^{h_{I,N}r_I}$ (we ignore the small probability that $\mathcal{A}$ correctly guess this value without making the corresponding $\mathcal{K}$ query – a standard argument in random oracle model). Now $\mathcal{S}$ randomly chooses one of $\mathcal{A}$'s $\mathcal{K}$'s queries π. If $\mathcal{S}$ is lucky enough that π is the above keying material (event VI), $\mathcal{S}$ answers the interactive BDH challenger correctly with $(\pi/(\hat{e}(xP, X + h(yP))^{h_{I,N}r_I})^{1/(r_I\alpha)}$.

Probability analysis:

I. If event V does not occur, neither does event I.
II. Let $N_{\mathcal{H}}$ be the number of $\mathcal{H}_0$ queries and k be the security parameter, collusion would not occur with probability $(2^k - N_{\mathcal{H}})/2^k$.
III. If event V does not occur, neither does event III.
IV. If event V does not occur, neither does event IV.
V. Let $N_{\mathcal{C}}$ be the number of sessions created, $\mathcal{A}$ chooses the session $\Pi^N_{I,J}$ with probability $1/N_{\mathcal{C}}$.
VI. Let $N_{\mathcal{K}}$ be the number of key derivation oracle queries, event VI occurs with probability $1/N_{\mathcal{K}}$.

$\mathcal{S}$ wins the game if event II and V does not occur but event VI occurs. If $\mathcal{A}$ is able to have an advantage $\epsilon(k)$ against our protocol, then $\mathcal{S}$ can also win with an advantage of at least $\frac{\epsilon(k)(2^k - N_{\mathcal{H}})}{N_{\mathcal{C}}N_{\mathcal{K}}2^k}$. However, since such an adversary $\mathcal{A}$ does not exist, the proof for Theorem 1 follows easily. □

Key compromise may lead to another problem. When the long-term key of an entity, A, is compromised; the adversary may be able to masquerade not only *as* A but also *to* A as another party, B. Our protocol is resistance to such attacks.

Theorem 2. *The protocol described in Figure 2 provides key compromise impersonation resilience (*KCIR*) assuming that the BDH problem is hard and $\mathcal{H}$, $\mathcal{H}_0$, and $\mathcal{K}$ are modelled as random oracles.*

Proof. Following the approaches of Chen and Kudla [9] and Krawczyk [22], we make a slight modification to the security model to capture KCIR – $\mathcal{A}$ is allowed to corrupt the initiating party, ID_I. The simulation by $\mathcal{S}$ will not abort even if $\mathcal{A}$ requested for the private key of ID_I. Therefore, the proof for Theorem 1 will not be invalidated by this change and Theorem 2 follows. □

4.3 Forward-Secrecy and Escrow-Freeness

Although an adversary can masquerade as the compromised entity once the latter's long-term key has been compromised, we do not want the adversary to also obtain previously accepted session keys. Protocols that prevent this are said to provide forward secrecy. As there is usually a computational cost in providing perfect forward secrecy, it is sometimes sacrificed and a weaker notion is considered. One example is *partial* forward secrecy whereby the compromise of *one* long-term private key or *both* ephemeral secrets of the communicating parties does not lead to the leakage of previously accepted session keys. No such protection is made when *both* parties' long-term keys are compromised. This notion is considered in existing ID-based protocols such as those of Chen and Kudla [9]. For our basic protocol, the proof of indistinguishability allows the adversary to ask Corrupt query for the ID_I associated with the test session, it follows that our protocol also achieve partial forward-secrecy.

There is an additional concern in forward secrecy for ID-based protocols when compared with those in conventional public key cryptography – the master secret of the KGC is another secret that can be compromised. When this happens, the long-term keys of all users will be compromised although it may be possible that no previously accepted session keys are deduced. Achieving this notion also mean that the key agreement protocol is *escrow-free*, assuming that there is no active attack by the KGC (e.g., by actively impersonating a user). A protocol is said to provide *KGC forward secrecy* (KGC-FS) if it retains confidentiality of previously accepted session keys even when the master secret of the KGC is compromised. It is easy to see that our protocol described in Figure 2 does not provide KGC-FS since any adversary with the knowledge of s will be able to compute $\hat{e}(W_A + h_A Q_A, W_B + h_B Q_B)^s = \hat{e}(Q_A, Q_B)^{s(a+h_A)(b+h_B)}$.

KGC-FS implies forward secrecy in the usual sense since all users' private keys can be computed with the master secret. It has been noted that two-party protocols with only two-message flow and having no previous establishment of secure shared state cannot achieve *perfect* forward secrecy [22]. Our protocol, having only two messages in the message flow, inherently cannot achieve *perfect* forward secrecy, not to say perfect KGC-FS. Here we consider *weak* KGC-FS, such that the previously established sessions without the active involvement of the adversary cannot be "recovered" even if the long-term key is compromised. We adopt the approach of Chen and Kudla [9] to give our protocol the same level of KGC-forward-secrecy as their protocol. The new protocol is described in Figure 3, with the underlined value indicates the changes from Figure 2.

Initiator A		Responder B
$a \in_R \mathbb{Z}_q^*; W_A := aQ_A; \underline{T_A := aP}$	$\xrightarrow{ID_A, W_A, \underline{T_A}}$	$b \in_R \mathbb{Z}_q^*; W_B := bQ_B; \underline{T_B := bP}$
		$h_A := \mathcal{H}_0(W_A, \underline{T_A}, \mathsf{ID}_B)$
		$sid_B := \mathsf{ID}_A \Vert W_A \Vert \underline{T_A} \Vert \mathsf{ID}_B \Vert W_B \Vert \underline{T_B}$
		$sk_B := \mathcal{K}(\hat{e}(W_A + h_A Q_A, (b + h_B)S_B), \underline{bT_A})$
$h_B := \mathcal{H}_0(W_B, \underline{T_B}, \mathsf{ID}_A)$	$\xleftarrow{ID_B, W_B, \underline{T_B}}$	
$sid_A := \mathsf{ID}_A \Vert W_A \Vert \underline{T_A} \Vert \mathsf{ID}_B \Vert W_B \Vert \underline{T_B}$		
$sk_A := \mathcal{K}(\hat{e}((a + h_A)S_A, W_B + h_B Q_B), \underline{aT_B})$		
$sk_A = \mathcal{K}(\hat{e}(Q_A, Q_B)^{s(a+h_A)(b+h_B)}, \underline{abP}) = sk_B$		

Fig. 3. Proposed escrow-free high-performance identity-based key agreement protocol

Informally, the protocol described in Figure 3 provides KGC-FS at the expense of two additional offline scalar-point multiplications and one online scalar-point multiplication. Learning s will not help the adversary in computing $\mathcal{K}(\hat{e}((a + h_A)Q_A, (b + h_B)Q_B)^s, \underline{abP})$ as finding abP means the CDH problem is solvable (since both a and b are deleted from the internal states upon completion of the protocol execution). Using the same exponent in the elements T and W allows a saving of one pseudorandom number generation and hence, faster exponentiation operation using the same exponent is possible. Security assurance is given by the following three theorems. Proofs are presented in the full paper [19].

Theorem 3. *The protocol described in Figure 3 is secure assuming that the Modified (Computational) Bilinear Diffie-Hellman (MBDH) problem is hard and $\mathcal{H}$, $\mathcal{H}_0$, and $\mathcal{K}$ are modelled as random oracles.*

Theorem 4. *The protocol described in Figure 3 provides weak* KGC-forward-secrecy *(*KGC-FS*) assuming that the Computational Diffie-Hellman (CDH) problem is hard and $\mathcal{H}$, $\mathcal{H}_0$, and $\mathcal{K}$ are modelled as random oracles.*

Theorem 5. *The protocol described in Figure 3 provides key compromise impersonation resistance assuming that the Modified Bilinear Diffie-Hellman (MBDH) problem is hard and $\mathcal{H}$, $\mathcal{H}_0$, and $\mathcal{K}$ are modelled as random oracles.*

4.4 Comparison with Existing Protocols

Table 1 describes the summary of comparison between several two-party ID-based protocols with two message flows. M denotes scalar-point multiplication, H denotes MapToPoint function [4] hashing identity to a point on an elliptic curve, and P denotes pairing in the table. Off-line computation can be pre-computed before the execution of the protocol, which includes public key derivation. Note that pairings are expensive and should be avoided whenever possible. MapToPoint is slightly more expensive but its cost is still comparable with that of scalar-point multiplication.

Table 1. Security and efficiency for two-party, two-message ID-based protocols

Protocol	Computation			Forward Secrecy	KCIR	Proof/ Attack
	On-line	Off-line	Public Key			
Our protocol #1	$1M + 1P$	$2M$	$1H$	FS	Yes	CK
Our protocol #2	$2M + 1P$	$3M$	$1H$	wKGC	Yes	CK
Wang [32]	$2M + 1P$	$1M$	$1H$	FS	Yes	BR
The following protocols are proven secure in a restricted model.						
Chen-Kudla #2 [9]	$1P$	$2M$	$1H$	No	Yes	wBR
Chen-Kudla #2' [9]	$1M + 1P$	$3M$	$1H$	wKGC	No	wBR
McCullagh-Barreto #1 [25]	$1P$	$2M$	$1M$	FS	No	wBR
McCullagh-Barreto #2 [25]	$1P$	$2M$	$1M$	? [5]	No	wBR[6]
The following protocols do not have any security proofs.						
Smart [30]	$1P$	$2M + 1P$	$1H$	No	Yes	No
Chen-Kudla #1' [9]	$1M + 1P$	$2M + 1P$	$1H$	wKGC	Yes	No
The following protocols are broken.						
Yi [34]	$1M + 1P$	$2M$	$1H$	See [19]		
Choie *et al.* #1 [13]	$1M + 2P$	$2M$	$1H$	See [5]		
Choie *et al.* #2 [13]	$2M + 1P$	$2M + 1P$	$1H$	See [5]		
Shim [27]	$1P$	$2M$	$1H$	See [31]		
Xie #1 [33]	$1P$	$3M$	$1M$	See [28]		
Xie #2 [33]	$1P$	$3M$	$1M$	See [28]		

The notation wBR denotes a restricted variant of the BR model whereby Session-Key Reveal query is not supported, FS denotes user *forward secrecy* while wKGC denotes *weak KGC forward secrecy*, and KCIR denotes key compromise impersonation resistance.

As shown in Table 1, among the "unbroken" ID-based protocols that provide:

KCIR **and** FS **(not** KGC-FS**).** Our protocol described in Figure 2 and Wang's protocol [32] are the most efficient. However, our protocol is based on a milder assumption and yet proven secure in a stronger model, which makes it more attractive than that of Wang's.

KCIR **and** KGC-FS**.** Although our protocol described in Figure 3 is a bit less efficient than that of Chen and Kudla [9] protocol #2', our protocol is proven secure in a stronger model (allowing the adversary to ask the Session-State Reveal query).

5 Ad-Hoc Anonymous Key Agreement Protocols

This section describes our extended protocol for ad-hoc anonymous key agreement based on the ID-based ring signature scheme of Chow *et al.* [20].

[5] No formal proof is given, it is unclear that whether the protocol can achieve anything stronger than weak KGC-FS.

[6] It is secure in the wBR model if the mistakes in its proof are corrected.[10,17].

5.1 Our Extension

In an ad-hoc anonymous key agreement protocol, the initiator conscripts a set of users – the *initiating ring* – and similarly the responder hides in a *responding ring*. Let A_j be a member of the initiating ring $A = \{A_1, A_2, \ldots, A_J\}$ and B_k be a member of the responding ring $B = \{B_1, B_2, \ldots, B_K\}$. Note that J can be different from K. For the security proof, we require each user to derive a value ψ in each session that is different from the values chosen in previous sessions with overwhelming probability. The values of A_j and B_k are denoted by ψ_A and ψ_B respectively. Canetti and Krawczyk suggested such a pair of (ψ_A, ψ_B) constitutes a unique session identifier for each session in practice[8].

1. A_j chooses $U_i \in_R \mathbb{G}$ and computes $h_i = \mathcal{H}_0(U_i, B, \psi_A), \forall i \in \{1, \ldots, J\} \setminus \{j\}$.
2. B_k then picks $V_i \in_R \mathbb{G}$, computes $c_i = \mathcal{H}_0(V_i, A, \psi_B), \forall i \in \{1, \ldots, K\} \setminus \{k\}$.
3. A_j chooses $r'_j \in_R \mathbb{Z}_q^*$, computes $U_j = r'_j Q_{A_j} - \sum_{i \neq j}\{U_i + h_i Q_{A_i}\}$.
4. Similarly, B_k chooses $r'_j \in_R \mathbb{Z}_q^*$, computes $V_k = r'_k Q_{B_k} - \sum_{i \neq k}\{V_i + c_i Q_{B_i}\}$.
5. A_j and B_k exchange $\bigcup_{i \in \{1,\ldots,J\}} \{U_i\}$ and $\bigcup_{i \in \{1,\ldots,K\}} \{V_i\}$
6. A_j and B_k compute session key sk_A and sk_B respectively as in (♠) and (♡).

$$
\begin{aligned}
sk_A &= \mathcal{K}(\hat{e}((r'_j + h_j)S_{A_j}, \sum_{i=1}^{K}(V_i + c_i Q_{B_i}))) \cdots (\spadesuit) \\
&= \mathcal{K}(\hat{e}(r'_j Q_{A_j} + h_j Q_{A_j}, \sum_{i=1}^{K}(V_i + c_i Q_{B_i}))^s) \\
&= \mathcal{K}(\hat{e}(U_j + \sum_{i \neq j}\{U_i + h_i Q_{A_i}\} + h_j Q_{A_j}, \sum_{i=1}^{K}(V_i + c_i Q_{B_i}))^s) \\
&= \mathcal{K}(\hat{e}(\sum_{i=1}^{J}(U_i + h_i Q_{A_i}), \sum_{i=1}^{K}(V_i + c_i Q_{B_i}))^s) \\
&= \mathcal{K}(\hat{e}(\sum_{i=1}^{J}(U_i + h_i Q_{A_i}), V_k + \sum_{i \neq k}\{V_i + c_i Q_{B_i}\} + c_k Q_{B_k})^s) \\
&= \mathcal{K}(\hat{e}(\sum_{i=1}^{J}(U_i + h_i Q_{A_i}), (r'_k + c_k)S_{B_k})) \\
&= \mathcal{K}(\hat{e}(\sum_{i=1}^{J}(U_i + h_i Q_{A_i}), r'_k Q_{B_k} + c_k Q_{B_k})^s) = sk_B \cdots (\heartsuit)
\end{aligned}
$$

5.2 Security Attributes

For simplicity, we assume both rings are of the same size, n. Apart from the conventional security properties for key agreement protocols, the security of ad-hoc anonymous key agreement protocols also depend on 1-out-of-n anonymity

[8] See [15] for a detail discussion on session identifier in key establishment protocols.

as described in Definition 3. These properties can be seen as a natural extension from the security requirements of key agreement protocol and those of ring signatures (e.g., see [20]).

Definition 3 (Security Attributes of Ad-Hoc Anonymous Key Agreement Protocols). *An ad-hoc anonymous key agreement protocol is secure if below conditions are satisfied.*

1: Validity. *If two uncorrupted oracles complete matching sessions, then both oracles must hold the same session key.*

2: Indistinguishability. *For all probabilistic, polynomial time adversaries, $\mathcal{A}$, the advantage of $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}(k)$, in game $\mathcal{G}$*[9] *is negligible. In particular, this implies* **1-out-of-n authenticity***: for all probabilistic, polynomial time adversaries, $\mathcal{A}$, without any one of the n private keys, has negligible advantage in learning about a fresh session key.*

3: 1-out-of-n Anonymity. *An ad-hoc anonymous key agreement protocol is said to have unconditional anonymity if for any group of n users, any adversary $\mathcal{A}$ (including the responder and the KGC) is unable to identify the real initiator better than a random guess, i.e., $\mathcal{A}$ can guess the identity of the initiator correctly with probability no better than $\frac{1}{n}$, or $\frac{1}{n-1}$ if $\mathcal{A}$ is in the ring. If the protocol satisfies* bilateral *privacy, the same requirement applies on the responding party.*

It is straightforward to see that our proposed protocol is valid. The indistinguishability and the 1-out-of-n anonymity properties are formally captured by Theorems 6 and 7 respectively. The proofs can be found in the full paper [19].

Theorem 6. *The protocol described in Section 5.1 achieves indistinguishability assuming that the Bilinear Diffie-Hellman (BDH) problem is hard and $\mathcal{H}$, $\mathcal{H}_0$, and $\mathcal{K}$ are modelled as random oracles.*

Theorem 7. *The protocol described in Section 5.1 provides 1-out-of-n anonymity unconditionally.*

We remark that it is also possible to equip this protocol with weak KGC forward secrecy by using the trick presented in Section 4.3. However, previously used ephemeral parameters should not be re-used for full-protection of the anonymity (since the exponent of the element V_i corresponding to the real identity is unknown, established a key using the knowledge of an exponent excludes one possibility for the real identity).

6 Conclusion and Future Work

In conclusion, we had proposed a new identity-based (ID-based) key agreement protocol, proven secure in the Canetti–Krawczyk model that allows the adversary access to the Session-Key Reveal and Session-State Reveal queries. Our protocol is

[9] Definition can be found in the Appendix of the full paper [19].

the *first* to be proven secure against such a strong adversary *without* employing any gap assumption. Using the approach of Chen and Kudla [9], we show how to provide KGC forward secrecy for our proposed ID-based protocol. As a result, both proposed protocols are efficient and yet proven secure in the strongest model among other previously published two-party two-message ID-based protocols with similar security attributes claim.

Motivated by the need for a better anonymous roaming mechanism and our observation that existing research appears to focus only on *unilateral* identity privacy, our basic protocol is extended to realize the first ad-hoc anonymous ID-based key agreement protocol with *bilateral* privacy.

Directions for future work include the following:

1. Our protocol only support the Session-State Reveal queries partially under the BDH assumption. We have seen examples of gap assumptions achieving a higher level of security. For example, the security proof of the Diffie–Hellman-based HMQV protocol is strengthened when the underlying assumption is changed from computational Diffie-Hellman assumption to its gap version [22]. It will be interesting to check if our protocol can also be strengthened by using the gap BDH assumption.
2. Finding more real-world applications for our proposed ID-based ad-hoc anonymous key agreement protocol.

References

1. Ateniese, G., Herzberg, A., Krawczyk, H., Tsudik, G.: Untraceable Mobility or How to Travel Incognito. Computer Networks 31(8), 871–884 (1999)
2. 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)
3. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 110–125. Springer, Heidelberg (1994)
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. SIAM Journal on Computing 32(3), 585–615 (2003)
5. Boyd, C., Choo, K.-K.R.: Security of Two-Party Identity-Based Key Agreement. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 229–243. Springer, Heidelberg (2005)
6. Boyd, C., Mao, W., Paterson, K.: Deniable Authenticated Key Establishment for Internet Protocols. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols. LNCS, vol. 3364, pp. 255–271. Springer, Heidelberg (2005)
7. Boyd, C., Park, D.: Public Key Protocols for Wireless Communications (Available from http://sky.fit.qut.edu.au/~boydc/papers/icisc98.ps.gz). In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 47–57. Springer, Heidelberg (2000)
8. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels (available from http://eprint.iacr.org/2001/040). In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)

9. Chen, L., Kudla, C.: Identity Based Authenticated Key Agreement Protocols from Pairings. In: CSFW 2003, pp. 219–233. IEEE Computer Society Press, Los Alamitos (2003), Corrected version at http://eprint.iacr.org/2002/184
10. Cheng, Z., Chen, L.: On Security Proof of McCullagh-Barreto's Key Agreement Protocol and its Variants. Cryptology ePrint Archive, Report 2005/201 (2005)
11. Cheng, Z., Chen, L., Comley, R., Tang, Q.: Identity-Based Key Agreement with Unilateral Identity Privacy Using Pairings. In: Chen, K., Deng, R., Lai, X., Zhou, J. (eds.) ISPEC 2006. LNCS, vol. 3903, pp. 202–213. Springer, Heidelberg (2006)
12. Cheng, Z., Nistazakis, M., Comley, R., Vasiu, L.: On the Indistinguishability-Based Security Model of Key Agreement Protocols-Simple Cases. Cryptology ePrint Archive, Report 2005/129 (2005)
13. Choie, Y.J., Jeong, E., Lee, E.: Efficient Identity-based Authenticated Key Agreement Protocol from Pairings. Applied Mathematics and Computation 162(1), 179–188 (2005)
14. Choo, K.-K.R.: Key Establishment: Proofs and Refutations. Ph.D. Thesis, Queensland University of Technology (2006), http://adt.library.qut.edu.au/adt-qut/public/adt-QUT20060928.114022/
15. Choo, K.-K.R.: A Proof of Revised Yahalom Protocol in the Bellare and Rogaway (1993) Model. The Computer Journal (2007) (Pre-print version available from http://eprint.iacr.org/2007/188
16. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
17. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: On Session Key Construction in Provably Secure Protocols. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 116–131. Springer, Heidelberg (2005)
18. Chow, S.S.M.: Personal Communication with Authors of [17] (April 29, 2005)
19. Chow, S.S.M., Choo, K.-K.R.: Strongly-Secure Identity-based Key Agreement and Anonymous Extension. Cryptology ePrint Archive, Report 2007/018. Full version of this paper (2007)
20. Chow, S.S.M., Yiu, S.M., Hui, L.C.K.: Efficient Identity Based Ring Signature. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 499–512. Springer, Heidelberg (2005)
21. Chow, S.S.M., Yiu, S.M., Hui, L.C.K., Chow, K.P.: Efficient Forward and Provably Secure ID-Based Signcryption Scheme. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004)
22. Krawczyk, H.: HMQV: A High-Performance Secure Diffie–Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
23. Kudla, C., Paterson, K.G.: Modular Security Proofs for Key Agreement Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–569. Springer, Heidelberg (2005)
24. Kunz-Jacques, S., Pointcheval, D.: About the Security of MTI/C0 and MQV. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 156–172. Springer, Heidelberg (2006)
25. McCullagh, N., Barreto, P.S.L.M.: A New Two-Party Identity-Based Authenticated Key Agreement. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 262–274. Springer, Heidelberg (2005)
26. Samfat, D., Molva, R., Asokan, N.: Untraceability in Mobile Networks. In: ACM MobiCom 1995, pp. 26–36. ACM Press, New York (1995)

27. Shim, K.-A.: Efficient ID-based Authenticated Key Agreement Protocol based on Weil Pairing. IEE Electronics Letters 39(8), 653–654 (2002)
28. Shim, K.-A.: Cryptanalysis of Two ID-based Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2005/357 (2005)
29. Shoup, V.: On Formal Models for Secure Key Exchange (Version 4). Technical Report RZ 3120 (#93166), IBM Research, Zurich (1999)
30. Smart, N.: An Identity based Authenticated Key Agreement Protocol based on the Weil Pairing. IEE Electronics Letters 38(13), 630–632 (2002)
31. Sun, H.-M., Hsieh, B.-T.: Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2003/113 (2003)
32. Wang, Y.: Efficient Identity-Based and Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/108 (2005)
33. Xie, G.: An ID-Based Key Agreement Scheme from Pairing. Cryptology ePrint Archive, Report 2005/093 (2005)
34. Yi, X.: Efficient ID-Based Key Agreement from Weil Pairing. IEEE Electronics Letters 39(2), 206–208 (2003)

Small Private-Exponent Attack on RSA with Primes Sharing Bits*

Yao-Dong Zhao and Wen-Feng Qi

Department of Applied Mathematics, Zhengzhou Information Engineering University, P.O. Box 1001-745, Zhengzhou, 450002, P.R. China
zhaoyaodong_1979@yahoo.com.cn, wenfeng.qi@263.net

Abstract. We show in this paper that if the primes share their some bits (e.g. Least-Significant bits), RSA system with small private-exponent is much more vulnerable to the Boneh-Durfee Attack.

1 Introduction

Since 1978, RSA public key cryptosystem has been widely used in various cryptosystems. However, it is computationally expensive whenever it is used for the encryption or the signature. So researchers are working hard to study how to speed up RSA algorithm and how to implement RSA system in the modern portable devices with low computational power which need to perform cryptographic operations, such as smart cards and mobile phones. Perhapes the simplest method to speed up RSA is shortening the public-exponent e or the private-exponent d. Shortening the public-exponent e may speed up the encryption, while shortening the private-exponent d may speed up the signature. And a small exponent may save the precious memory of these devices.

In 1990, Wiener [14] showed that if the private-exponent $d < N^{0.25}$, one may disclose d in polynomial time from e and N, where $N = p \cdot q$ denotes the public modulus in RSA cryptosystem, p, q are two large primes. In 1999, Boneh and Durfee [1] presented an improvement which made the upper bound up to 0.292. That means if $d < N^{0.292}$, the private-exponent d may be found in polynomial time. Their method is based on the lattice-based work by Don Coppersmith [2]-[4] to find small foots to low-degree bivariate modular polynomial equations. The method is heuristic, however the attack works very well in practice.

It is well known that small prime difference makes RSA insecure. For example, in 2002, Weger showed in [13] if $|p - q| = \Delta$ is small, RSA system with small exponent is much more vulnerable. So some standards require a certain condition on the difference of the primes. For example, ANSI X9.13 requires that the two primes differ in the first 100 bits. But in this paper we show that only requiring the primes to have large difference (e.g. the Most-Significant bits of the primes

* This work was supported by the National Natural Science Foundation of China (Grant 60673081) and "863" project of China (Grant $2006AA01Z417$).

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 221–229, 2007.

are not equal) will not ensure the security of RSA with small private-exponent. If the primes share their Least-Significant bits (For example, the system proposed in [10]-[11], but the private-exponent used in this system is not small), RSA system with small private-exponent is much more vulnerable to the Boneh-Durfee Attack. Let $N = p \cdot q$ denote an RSA modulus of length n bits, with p and q primes each of length about $n/2$ bits. Suppose $|p - q| = r \cdot 2^{(1/2-\alpha)\cdot n}$, and the public-exponent e in the RSA system is a $\gamma \cdot n$-bit odd positive integer. Let the private-exponent d be a $\beta \cdot n$-bit positive integer. Our result is if

$$\beta < (\alpha + 13/2 - (4\alpha^2 + 4\alpha + 24\alpha\gamma + 12\gamma + 1)^{1/2})/6,$$

one may disclose the private-exponent d in polynomial time. For example, when p and q share their $0.24n$ Least-Significant bits RSA will be insecure if $d < N^{0.371}$.

The attack is similar to [5]. We modify the lattice which will be used to solve the prime factors of the RSA modulus N. And the attack is also heuristic, however the attack works very well in practice. If the length of r is much less than αn, that means p and q share their Most-Significant bits, our result is much worse than the result in [13]. However, using the method in [13], one can not get results better than the result in this paper when the length of r is very close to αn. So our result also shows that choosing an RSA modulus with its prime factors sharing some bits yields improvements on the small private-exponent attack of Boneh-Durfee.

The paper is organized as follows. Some preliminary facts in lattice theory and the basic steps of the attack are given in Section 2; In Section 3, we present the attack on the RSA system with primes sharing least significant bits and small private-exponent. The conclusion and some problems are proposed in Section 4.

2 Some Facts of Lattice and Lattice Attack

Let L be a lattice spanned by linearly independent vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w$, where $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w \in \mathbb{Z}^n$ and $w \leq n$. That is, $L = \{\sum_{i=1}^{w} k_i \mathbf{u}_i | k_i \in \mathbb{Z}\}$. We say that the set $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w\}$ is a basis for L, and w is the dimension of L. We denote the vectors by $\mathbf{u}_1^*, \mathbf{u}_2^*, \ldots, \mathbf{u}_w^*$ which are obtained by applying the Gram-Schmidt process to the vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w$. Then the determinant of L is defined as

$$\det(L) = \prod_{i=1}^{w} ||\mathbf{u}_i^*||,$$

where $|| \mathbf{a} ||$ denotes the Euclidean norm of the vector $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$, that is,

$$|| \mathbf{a} || = (\sum_{i=0}^{n-1} a_i^2)^{1/2}.$$

If $w = n$, the lattice is called a full rank lattice. A full rank lattice has the property that

$$\det(L) = |\det(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w)|,$$

where $|\det(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w)|$ is the absolute value of the determinant of the $n \times n$ matrix whose rows are the basis vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w$.

It is well known that given a basis for a lattice, LLL algorithm can provide an approximately shortest vector in the lattice [8].

Lemma 1 ([8]). *Let linearly independent vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_w$ be the inputs of LLL algorithm, and $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_w$ the output vectors. Then $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_w$ satisfy*

1) $||\mathbf{b}_i^*||^2 < 2||\mathbf{b}_{i+1}^*||^2$,

2) *for $i = 1, \ldots, w$, if $\mathbf{b}_i = \mathbf{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*$, then $\mid \mu_{i,j} \mid < 1/2$.*

Remark 1. If the lattice L is spanned by $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_w$, then the set of vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_w$ is called an LLL-reduced basis of L.

Lemma 2 ([1]). *Let L be a lattice, $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_w$ an LLL-reduced basis of L. Then*

$$||\mathbf{b}_1|| \leq 2^{w/2} det(L)^{1/w},$$
$$||\mathbf{b}_2|| \leq 2^{w/2} det(L)^{1/(w-1)}.$$

In 1996, Don Coppersmith found the method based on LLL algorithm to solve the small roots of small-degree modular equations [2]-[4]. This method was simplified by Howgrave-Graham [6]. The simplified method is based on the following lemma.

Let $h(x, y, z) = \sum_{i,j,k} a_{i,j,k} x^i y^j z^k$ be a trivariate polynomial. Then the norm of this polynomial is defined as

$$||h(x, y, z)|| = (\sum_{i,j,k} a_{i,j,k}^2)^{1/2}.$$

Lemma 3 ([6]). *Let $h(x, y, z) = \sum_{i,j,k} a_{i,j,k} x^i y^j z^k$ be a trivariate polynomial, $a_{i,j,k} \in \mathbb{Z}$, which is a sum of w monomials. If*

1) *$h(x_0, y_0, z_0) \equiv 0 \mod N$, where $x_0 < X$, $y_0 < Y$, $z_0 < Z$, $X, Y, Z, N \in \mathbb{Z}$,*

2) *$||h(xX, yY, zZ)|| < N/(w^{1/2})$,*

then $h(x_0, y_0, z_0) = 0$.

Lemma 3 tells us that if the modular polynomial equation has small roots and the condition (2) can be satisfied, then the problem to find these roots can be convert to the problem to solve the polynomial equation over rational number field.

Suppose $f(x_0, y_0, z_0) = 0 \mod e$, and $g(y_0, z_0) = 0$ for some small integers $x_0, y_0, z_0 \in \mathbb{Z}$, and $f(x, y, z), g(y, z)$ are two polynomials over $\mathbb{Z}$. Construct the set of polynomials

$$h_{u_1,u_2,u_3,v}(x, y, z) = e^{m-v} x^{u_1} y^{u_2} z^{u_3} f^v(x, y, z),$$

for u_1, u_2, u_3, v, m, $u_1 \in I_1$, $u_2 \in I_2$, $u_3 \in I_3$, $v \in I$ where I_1, I_2, I_3, I are the sets of some positive integers. It is obvious that $h_{u_1,u_2,u_3,v}(x_0, y_0, z_0) = 0 \bmod e^m$. Then the linear combination $h(x, y, z)$ of these polynomials has $h(x_0, y_0, z_0) = 0 \bmod e^m$. If $h(x, y, z)$ satisfies the second condition of Lemma 3, that is $||h(xX, yY, zZ)|| < e^m/(w^{1/2})$, where X, Y, Z are the upper bound of x_0, y_0, z_0 and w is the number of the monomials in $h(x, y, z)$, then $h(x_0, y_0, z_0) = 0$ from Lemma 3. If we have many polynomials satisfying Lemma 3, then we may get the root (x_0, y_0, z_0) by the resultant of these polynomials.

Using LLL algorithm, the polynomials with low norm may be found. We may construct a lattice L spanned by the corresponding coefficients of the sets of polynomials $h_{u_1,u_2,u_3,v}(xX, yY, zZ)$. Let r be the dimension of the lattice L. Then the polynomials corresponding to the first vector and the second vector of the output of LLL algorithm satisfy:

$$||h_1(xX, yY, zZ)|| < 2^{r/2}\det(L)^{1/r};$$
$$||h_2(xX, yY, zZ)|| < 2^{r/2}\det(L)^{1/(r-1)}.$$

If $2^{r/2}\det(L)^{1/(r-1)} < e^m/(r^{1/2})$, we have $h_1(x_0, y_0, z_0) = h_2(x_0, y_0, z_0) = 0$ from Lemma 3. The factor $2^{r/2}$ could be negligible since it is too small compared to e^m. So in order to make the conditions in Lemma 3 satisfied, we only need

$$\det(L) < e^{m(r-1)}.$$

If the above inequation is satisfied, we may have $h_1(x_0, y_0, z_0) = 0$ and $h_2(x_0, y_0, z_0) = 0$. Let $g'(y, z) = \mathrm{Res}_x(h_1, h_2)$ be the resultant of the polynomials $h_1(x, y, z)$ and $h_2(x, y, z)$. If $g'(y, z) \neq 0$, then $\mathrm{Res}_z(g', g)$ may disclose the root y_0.

In the above discussion, we suppose the resultant of the polynomials is not equal to 0. So we always have the following assumption.

Assumption 1. *The resultant computations for the polynomials in this paper yield non-zero polynomials.*

3 The Attack

Let $N = p \cdot q$ be an n-bit positive integer, where p, q are two $n/2$-bit primes which share their $(1/2 - \alpha)n$ least significant bits. Suppose the public-exponent e in the RSA system is a $\gamma \cdot n$-bit odd positive integer, and the private-exponent d is a $\beta \cdot n$-bit positive integer. Then from the relationship between e and d, there exists a positive integer k satisfying

$$ed = k(N + 1 - (p + q)) + 1.$$

Assume $p - q = r2^{(1/2-\alpha)n}$, where r is an αn-bit positive integer. Then we have

$$ed = k(N + 1 - (r2^{(1/2-\alpha)n} + 2q)) + 1.$$

Let $A = N + 1$, $a = 2^{(1/2-\alpha)n}$, then (k, q, r) is a root for the modular polynomial

$$f(x, y, z) = x(A - 2y - az) + 1 \bmod e.$$

So if we can solve the equation

$$f(x_0, y_0, z_0) = 0 \bmod e,$$

then we may disclose the private key d, where $|x_0| < X, |y_0| < Y, |z_0| < Z$ and

$$X \cong ed/N \cong N^{\gamma+\beta-1}, Y \cong N^{1/2}, Z \cong N^{\alpha}.$$

Construct the set of polynomials as

$$g_{k,i,b}(x, y, z) = e^{m-k} x^i y^b f^k(x, y, z), \quad k = 0; \ldots, (m-1), i = 1, \ldots, (m-k); b = 0, 1;$$

$$h'_{k,j}(x, y, z) = e^{m-k}(az)^j f^k(x, y, z), \qquad k = 0, \ldots, m; j = 0, \ldots, t;$$

$$h''_{k,j}(x, y, z) = e^{m-k} y^j f^k(x, y, z), \qquad k = 0, \ldots, m; j = 1, \ldots, t;$$

where m and t are two parameters.

Remark 2. Since $y^2 + ayz = N$, any term such as $x^i y^j z^k$ could be replaced by the linear function on $x^i y^j$ and $x^i z^k$. So in the lattice we constructe, the term $x^i y^j z^k$ does not exist.

Let L denote the lattice spanned by the coefficient vectors of the polynomials $g_{k,i,b}(xX, yY, zZ), h'_{k,j}(xX, yY, zZ), h''_{k,j}(xX, yY, zZ)$. Using LLL algorithm, we may get two vectors h_1, h_2, and $h_1(x_0, y_0, z_0)$, $h_2(x_0, y_0, z_0)$ denote the corresponding polynomials. Then

$$h_1(x_0, y_0, z_0) = 0 \bmod e^m; h_2(x_0, y_0, z_0) = 0 \bmod e^m,$$

where $x_0 = k, y_0 = q, z_0 = r$. If these two polynomials satisfy the second condition in Lemma 3, then $h_1(x_0, y_0, z_0) = 0$ and $h_2(x_0, y_0, z_0) = 0$. Since $az = N/y - y$, We may have two polynomials $H_1(x, y), H_2(x, y)$ with $H_1(x_0, y_0) = 0$ and $H_2(x_0, y_0) = 0$, where $(x_0, y_0) = (k, q)$. From the resultant $h(y) = \mathrm{Res}_x(H_1, H_2)$, we may get y_0, that is we may have the factor of the modulus N.

The problem now is when the polynomials the LLL algorithm outputs satisfy the second condition in Lemma 3. From the discussion above, we have the condition in Lemma 3 will be satisfied if the following inequation holds,

$$\det(L) < e^{m(w-1)}, \tag{1}$$

where w is the dimension of the lattice L.

From the construction of the lattice L, we have

$$\begin{aligned} \det(L) &= (ex)^{num_x} Y^{num_y} Z^{num_z} \\ &= N^{(2\gamma+\beta-1)num_x + 1/2 num_y + \alpha num_z}, \end{aligned}$$

where

$$num_x = 1/6m(m+1)(4m+3t+5) + 1/2m(m+1)t;$$
$$num_y = 1/6m(m+1)(m+2) + t/2 + mt + m^2t/2 + t^2/2 + mt^2/2;$$
$$num_z = 1/6(m^3-m) + 1/2t(1+m) + 1/2tm(m+1) + 1/2m(m+1) + 1/2t^2(m+1).$$

Then Inequation (11) will be satisfied when

$$(2\gamma+\beta-1)num_x + 1/2num_y + \alpha num_z < \gamma m((m+1)(m+2t+1)-1)$$

from $w = (m+1)(m+2t+1)$.

Suppose $t = \tau m$, then

$$num_x = (4+6\tau)/6m^3 + O(m^3);$$
$$num_y = 1/6(1+3\tau+3\tau^2)m^3 + O(m^3);$$
$$num_z = 1/6(1+3\tau+3\tau^2)m^3 + O(m^3).$$

So Inequation (11) is true if and only if

$$(2\gamma+\beta-1)((4+6\tau)/6m^3 + O(m^3)) + 1/2(1/6(1+3\tau+3\tau^2)m^3 + O(m^3)) + \alpha(1/6(1+3\tau+3\tau^2)m^3 + O(m^3)) < \gamma((1+2\tau)m^3 + O(m^3)).$$

Suppoes m is large enough, then the inequation above will be true if

$$(4\gamma+8\beta+2\alpha-7) + (12\beta+6\alpha-9)\tau + (6\alpha+3)\tau^2 < 0.$$

The left side of the inequation above has its least value in

$$\tau = -(6\alpha+12\beta-9)/(6+12\alpha).$$

From easy computation, we have the Inequation (11) will be true if

$$\beta < (\alpha + 13/2 - (4\alpha^2+4\alpha+24\alpha\gamma+12\gamma+1)^{1/2})/6,$$

We describe the relationship between the bound of β and the parameters α, γ by two charts and two tables. Chart 1 describes the bound of β when $\gamma = 1$; Chart 2 describes the bound of β when $\gamma = 0.55$. In Table 2, the data in the first row are the value of γ, and the data in the first column are the value of α. The data in the other positions are the value of the upper bound of β when γ and α are evaluated with the corresponding values.

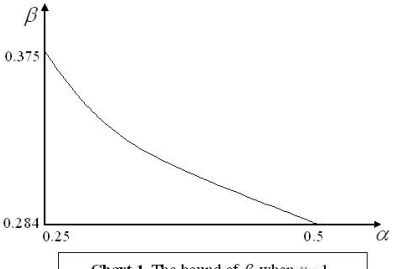


Chart 1. The bound of β when $\gamma = 1$.

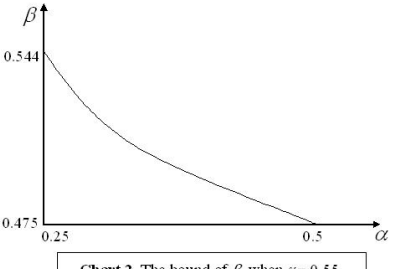


Chart 2. The bound of β when $\gamma = 0.55$.

Table 1. The matrix with $m = 3$, $t = 1$

	x	xy	x^2	x^2y	x^3	x^3y	x^2z	x^2y^2	x^3z	x^3y^2	x^3z^2	x^3y^3	1	y	z	xz	xy^2	xz^2	x^2z^2	x^2y^3	x^2z^3	x^3z^3	x^3y^4	x^3z^4
e^3x	e^3X																							
e^3xy		e^3XY																						
e^3x^2			e^3X^2																					
e^3x^2y				e^3X^2Y																				
e^3x^3					e^3X^3																			
e^3x^3y						e^3X^3Y																		
e^2xf	-		-	-			e^2X^2Z																	
e^2xyf		-	-	-				$e^2X^2Y^2$																
e^2x^2f			-		-	-			e^2X^3Z															
e^2x^2yf				-	-	-				$e^2X^3Y^2$														
exf^2	-		-	-	-	-	-		-		eX^3Z^2													
$exyf^2$		-	-	-	-	-		-	-	-		eX^3Y^3												
e^3													e^3											
e^3y														e^3Y										
e^3z															e^3Z									
e^2f	-	-											-			e^3XZ								
e^2yf	-	-											-				e^3XY^2							
e^2azf	-													-	-	-		e^2XZ^2						
ef^2	-	-	-	-									-		-				eX^2Z^2					
eyf^2	-	-	-	-			-	-					-			-				eX^2Y^3				
$eazf^2$	-		-				-	-						-	-	-	-	-			eX^2Z^3			
f^3	-	-	-	-	-	-	-		-		-	-	-		-			-				X^3Z^3		
yf^3	-	-	-	-	-	-	-	-	-	-	-	-	-			-			-				X^3Y^4	
azf^3	-		-		-		-	-	-	-	-			-	-	-	-	-		-	-	-		X^3Z^4

Table 2. The attack results when p and q share their least significant bits

	1.0	0.9	0.86	0.8	0.7	0.6	0.55
0.5	0.284	0.323	0.339	0.363	0.406	0.451	0.475
0.4	0.319	0.356	0.371	0.395	0.435	0.479	0.501
0.3	0.335	0.390	0.405	0.427	0.466	0.507	0.529
0.25	0.375	0.409	0.423	0.444	0.482	0.522	0.544

In this matrix, the entries in the first row correspond the terms appeared in the polynomials. And the entries in the first column is the polynomials used to construct the lattice. The symbol "-" means the coefficient of the corresponding term is not 0. The terms in the diagonal are the coefficients of the corresponding terms. And the other blanks in this matrix are 0. It is obvious that the matrix is lower triangular.

We performed several experiments to test the validity of Assumption 1. We implemented the new attacks on a 3.0GHz Pentium running Microsoft. The LLL lattice reduction was done using Mathematica 5.1. All the experiments can disclose the factors of the public modulus N.

4 Conclusion and Open Problems

In this paper, we show that if the primes share their some bits (e.g. Least-Significant bits), RSA system with small private-exponent is much more vulnerable to the Boneh-Durfee Attack. The result shows that we should be careful when we choose the primes. It is obvious that when $\alpha = 0.5$ and $\gamma = 1$, the upper bound of the private-exponent in our attack scheme d is $N^{0.284}$ which is lower than the result in [1]. How to modify the attack scheme to get the best bound is an open problem.

References

1. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans. on Information Theory 46(4), 1339–1349
2. Coppersmith, D.: Finding a Small Root of a Univariate Modular Equation. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 155–165. Springer, Heidelberg (1996)
3. Coppersmith, D.: Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (1996)
4. Coppersmith, D.: Finding Small Solutions to Small Degree Polynomials. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 178–189. Springer, Heidelberg (2001)
5. Durfee, G., Nguyen, P.Q.: Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiscrypt'99. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 14–29. Springer, Heidelberg (2000)

6. Howgrave-Graham, N.: Finding Small Roots of Univariate Modular Equations Revisited. In: Darnell, M. (ed.) Cryptography and Coding. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
7. Jochemsz, E., May, A.: A Strategy for Finding Roots of Multivariate Polynomials with New Applications in attacking RSA Variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006)
8. Lenstra, A., Lenstra, H., Lovasz, L.: Factoring Polynomials with Rational Coefficients. Mathematiche Annalen 261, 515–534
9. May, A.: New RSA Vulnerabilities Using Lattice Reduction Methods, Doctor Thesis
10. Steinfeld, R., Zheng, Y.: An advantage of Low-Exponent RSA with Modulus Primes Sharing Least Significant Bits. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 52–62. Springer, Heidelberg (2001)
11. Steinfeld, R., Zheng, Y.: On the security of RSA with Primes Sharing Least Significant Bits. AAECC 15(3), 179–200 (2004)
12. Sun, H.M., Yang, W.C., Laih, C.S.: On the design of RSA with short secret exponent. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 150–164. Springer, Heidelberg (1999)
13. Weger, B.D.: Cryptanalysis of RSA with Small Prime Difference. AAECC, vol. 13, pp. 17–28. Springer, Heidelberg
14. Wiener, M.: Cryptanalysis of Short RSA Secret Exponents. IEEE Trans. on Information Theory 36, 553–558

Multiple Modular Additions and Crossword Puzzle Attack on NLSv2

Joo Yeon Cho and Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography,
Department of Computing, Macquarie University,
NSW, Australia, 2109
{jcho,josef}@ics.mq.edu.au

Abstract. NLS is a stream cipher which was submitted to the eSTREAM project. A linear distinguishing attack against NLS was presented by Cho and Pieprzyk, which was called Crossword Puzzle (CP) attack. NLSv2 is a tweak version of NLS which aims mainly at avoiding the CP attack. In this paper, a new distinguishing attack against NLSv2 is presented. The attack exploits high correlation amongst neighboring bits of the cipher. The paper first shows that the modular addition preserves pairwise correlations as demonstrated by existence of linear approximations with large biases. Next, it shows how to combine these results with the existence of high correlation between bits 29 and 30 of the S-box to obtain a distinguisher whose bias is around 2^{-37}. Consequently, we claim that NLSv2 is distinguishable from a random cipher after observing around 2^{74} keystream words.

Keywords: Distinguishing Attacks, Crossword Puzzle Attack, Stream Ciphers, eSTREAM, NLS, NLSv2.

1 Introduction

In 2004, ECRYPT project launched a new multi-year project eSTREAM, the ECRYPT Stream Cipher project, to identify new stream ciphers that might become suitable for widespread adoption as international industry standards [8]. NLS is one of stream ciphers submitted to the eSTREAM project [4]. The second phase of the eSTREAM included NLS in both profiles 1 (Software) and 2 (Hardware). During the first phase, a distinguishing attack against NLS was presented in [3]. The attack requires around 2^{60} keystream observations.

NLSv2 is a tweaked version of NLS to counter the distinguishing attack mentioned above. Unlike in the original NLS, NLSv2 periodically updates the value $Konst$ every 65537 clock. The new value of $Konst$ is taken from the output of the non-linear filter. In [3], the linear approximation from non-linear feedback shift register (NFSR) was derived and the sign of bias can be either positive or negative depending on the value of $Konst$. Thus, a randomly updated $Konst$ is expected to "neutralize" the overall bias of approximations, which eventually minimizes the bias of distinguisher.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 230–248, 2007.

In [2], the authors presented distinguishing attacks on NLS and NLSv2 by Crossword Puzzle attack (or shortly CP attack) method. The CP attack is a variant of the linear distinguishing attack which was specifically designed to work for NFSR based stream ciphers. The attack concentrates on finding approximations and combining them in such a way that the internal states of NFSR cancel each other.

Being more specific, the authors showed that, for the attack on NLSv2, the effect of *Konst* could be eliminated by using 'even' number of NFSR approximations. A distinguisher was constructed by combining eight NFSR approximations and two NLF approximations, for which 2^{96} observations of keystream are required. However, due to the explicit upper limit of 2^{80} on the number of observed keystream imposed by the designers of the cipher, this attack does not break the cipher.

In this paper, we have improved the linear distinguishing attack on NLSv2 presented in the latter part of [2]. We still use the CP attack from [2] for our distinguisher. However, we have observed that there are linear approximations of S-boxes whose biases are much higher than those used in the previous attack. Using those more effective approximations, we can now construct a distinguisher whose bias is around 2^{-37}. Therefore, we claim that NLSv2 is distinguishable from a truly random cipher after observing around 2^{74} keystream words which are within the limit of permitted observations during the session with a single key.

This paper is organized as follows. Section 2 presents some properties of multiple modular additions which are useful for our attack. Section 3 presents the structure of NLSv2. Section 4 presents the technique we use to construct linear approximations required in our attack. Section 5 contains the main part of the paper and presents the CP attack against NLSv2. Section 6 concludes the work.

Notation

1. $\boxplus$ denotes the addition modulo 2^{32},
2. $x^{\lll k}$ represents the 32-bit x which is rotated left by k-bit,
3. $x_{(i)}$ stands for i-th bit of the 32-bit string x

These notations will be used throughout this paper.

2 Probabilistic Properties of Multiple Modular Additions

The attack on NLSv2 explores a correlation between two neighboring bits. This section describes the behavior of neighboring bits in modular additions and establishes the background for our study. Suppose that $z = x \boxplus y$ where $x, y \in \{0,1\}^{32}$ are uniformly distributed random variables. According to [1], each $z_{(i)}$ bit is expressed a function of $x_{(i)}, \cdots, x_{(0)}$ and $y_{(i)}, \cdots, y_{(0)}$ bits as follows

$$z_{(i)} = x_{(i)} \oplus y_{(i)} \oplus x_{(i-1)}y_{(i-1)} \oplus \sum_{j=0}^{i-2} x_{(j)}y_{(j)} \prod_{k=j+1}^{i-1} [x_{(k)} \oplus y_{(k)}], \quad \text{for } i = 1, \ldots, 31$$

and $z_{(0)} = x_{(0)} \oplus y_{(0)}$. Let $R(x, y)$ denote the carry of modular addition as follows

$$R(x,y)_{(i)} = x_{(i)}y_{(i)} \oplus \sum_{j=0}^{i-1} x_{(j)}y_{(j)} \prod_{k=j+1}^{i} [x_{(k)} \oplus y_{(k)}], \quad i = 0, 1, \ldots, 30. \tag{1}$$

Then, obviously, $z_{(i)} = x_{(i)} \oplus y_{(i)} \oplus R(x,y)_{(i-1)}$ for $i = 1, \ldots, 31$. Due to Equation (1), the carry $R(x,y)_{(i)}$ has the following recursive relation.

$$R(x,y)_{(i)} = x_{(i)}y_{(i)} \oplus (x_{(i)} \oplus y_{(i)})R(x,y)_{(i-1)} \tag{2}$$

Hereafter, we study the biases of approximations using a pair of adjacent bits when multiple modular additions are used. For this, we introduce the following definition.

Definition 1. *Γ_i denotes a linear masking vector over $GF(2)$ which has '1' only on the bit positions of i and $i+1$. Then, given 32-bit x, $\Gamma_i \cdot x = x_{(i)} \oplus x_{(i+1)}$, where $\cdot$ denote the standard inner product.*

Now we are ready to present a collection of properties that are formulated in the lemmas given below. These results are essential for setting up our attack. In the following, we assume that all inputs of modular addition are uniformly distributed random variables.

Lemma 1. *Given $x, y \in \{0,1\}^{32}$, the probability distribution of the carry bits can be expressed as follows*

$$Pr[R(x,y)_{(i)} = 0] = \frac{1}{2} + 2^{-i-2} \quad \textit{for } i = 0, \ldots, 30.$$

Proof. The proof is given by induction.

(1) Let $i = 0$. Then $Pr[R(x,y)_{(0)} = x_{(0)}y_{(0)} = 0] = \frac{3}{4} = \frac{1}{2} + 2^{-2}$
(2) In the induction step we assume that $Pr[R(x,y)_{(i-1)} = 0] = \frac{1}{2} + 2^{-i-1}$. Then, from Relation (2), we have

$$Pr[R(x,y)_{(i)} = 0] = \begin{cases} Pr[x_{(i)}y_{(i)} = 0] = \frac{3}{4}, & \text{if } R(x,y)_{(i-1)} = 0 \\ Pr[x_{(i)}y_{(i)} \oplus (x_{(i)} \oplus y_{(i)}) = 0] = \frac{1}{4}, & \text{if } R(x,y)_{(i-1)} = 1 \end{cases}$$

Hence, the following equation holds

$$Pr[R(x,y)_{(i)} = 0] = \frac{3}{4} \cdot Pr[R(x,y)_{(i-1)} = 0] + \frac{1}{4} \cdot Pr[R(x,y)_{(i)} = 1] = \frac{1}{2} + 2^{-i-2}.$$

This proves our lemma. □

Corollary 1. *Given $x, y \in \{0,1\}^{32}$, the following approximation holds with the constant probability*

$$Pr[\Gamma_i \cdot R(x,y) = 0] = \frac{3}{4} \quad \textit{for } i = 0, \ldots, 30.$$

Proof. By definition, we obtain

$$\Gamma_i \cdot R(x,y) = R(x,y)_{(i)} \oplus R(x,y)_{(i+1)} = x_{(i+1)} y_{(i+1)} \oplus (x_{(i+1)} \oplus y_{(i+1)} \oplus 1) R(x,y)_{(i)}.$$

Hence, from Lemma 1, we get

$$Pr[\Gamma_i \cdot R(x,y) = 0] = \frac{3}{4} \cdot Pr[R(x,y)_{(i)} = 0] + \frac{3}{4} \cdot Pr[R(x,y)_{(i)} = 1] = \frac{3}{4}$$

and the corollary holds. □

Due to Corollary 1, the following approximation has the probability of $\frac{3}{4}$, as stated in [2],

$$\Gamma_i \cdot (x \boxplus y) = \Gamma_i \cdot (x \oplus y), \quad i = 0, \ldots, 30 \tag{3}$$

Lemma 2. *Suppose that $x, y, z \in \{0,1\}^{32}$. Then, the following linear approximation*

$$\Gamma_i \cdot (x \boxplus y \boxplus z) = \Gamma_i \cdot (x \oplus y \oplus z) \tag{4}$$

holds with the probability of $\frac{2}{3} - \frac{1}{3} 2^{-2i-1}$ for $i = 0, \ldots, 30$.

Proof. The proof of the lemma can be found in Appendix A.

It is interesting to see that the probability of Approximation (4) is around $\frac{2}{3} = \frac{1}{2}(1 + 2^{-1.58})$ due to the dependency between the two modular additions. In contrast to Lemma 2, the approximation $\Gamma_i \cdot [(x \boxplus y) \oplus (z \boxplus w)] = \Gamma_i \cdot [(x \oplus y) \oplus (z \oplus w)]$ holds with the bias of $(2^{-1})^2$ by Piling-Up Lemma [6] since the two modular additions are mutually independent. A similar observation that linear approximations over two consecutive modular additions are not independent was exploited to construct an improved distinguisher for SNOW 2.0 in [9]. We note that Approximation (3) and Lemma 2 can be also derived by more generalized algorithm on the linear approximations of modular addition with any inputs given in [9].

Lemma 3. *Suppose that $x_1, x_2, \ldots, x_n \in \{0,1\}^{32}$ and $k \in \{0,1\}^{32}$ where n is an even number. Then, the following linear approximation*

$$\Gamma_i \cdot (x_1 \boxplus k) \oplus \Gamma_i \cdot (x_2 \boxplus k) \oplus \cdots \oplus \Gamma_i \cdot (x_n \boxplus k) = \Gamma_i \cdot (x_1 \oplus x_2 \oplus \cdots \oplus x_n)$$

holds with the probability of around $\frac{n+2}{2(n+1)}$ for $i = 1, \ldots, 30$.

Proof. The lemma is proved in Appendix B.

Corollary 2. *Given $x, y, z \in \{0,1\}^{32}$, the following linear approximation*

$$\Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (x \boxplus z) = \Gamma_i \cdot (y \oplus z)$$

holds with the probability of $\frac{2}{3} + \frac{1}{3} 2^{-2i-2}$ for $i = 0, \ldots, 30$.

Proof. Appendix C contains the proof of the Corollary.

Lemma 4. *Given $x, y, z, w \in \{0,1\}^{32}$, the following linear approximation*

$$\Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (z \boxplus w) = \Gamma_i \cdot (x \boxplus z) \oplus \Gamma_i \cdot (y \boxplus w)$$

has the probability of $\frac{2}{3} + \frac{1}{3}2^{-2i-2}$ for $i = 0, \ldots, 30$.

Proof. For the proof, see Appendix D.

Corollary 3. *Let $x, y, z, w \in \{0,1\}^{32}$, then the following linear approximation*

$$\Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (x \boxplus z) \oplus \Gamma_i \cdot (y \boxplus w) = \Gamma_i \cdot (z \oplus w)$$

holds with the probability of $\frac{29}{48} + \frac{1}{3}2^{-2i-4}$ for $i = 0, \ldots, 30$.

Proof. For the proof, see Appendix E.

For convenience, in the rest of the paper we are going to use a bias of approximation rather than the probability that an approximation holds.

3 Brief Description of NLSv2

NLS is a synchronous, word-oriented stream cipher controlled by a secret key of the size up to 128 bits. The keystream generator of NLS is composed of a non-linear feedback shift register (NFSR) and a non-linear filter (NLF) with a counter. In this section, we describe only the part of NLS which is necessary to understand our attack. The structure of NLSv2 is exactly the same as that of NLS except a periodically updated *Konst* [4]. For more details, we refer to the paper [5].

3.1 Non-linear Feedback Shift Register (NFSR)

At time t, the state of NFSR is denoted by $\sigma_t = (r_t[0], \ldots, r_t[16])$ where $r_t[i]$ is a 32-bit word. *Konst* is a key-dependent 32-bit word, which is set at the initialization stage and is updated periodically. The transition from the state σ_t to the state σ_{t+1} is defined as follows:

(1) $r_{t+1}[i] = r_t[i+1]$ for $i = 0, \ldots, 15$;
(2) $r_{t+1}[16] = f((r_t[0] \lll 19) \boxplus (r_t[15] \lll 9) \boxplus Konst) \oplus r_t[4]$;
(3) if $t \equiv 0$ (modulo f16), then
   (a) $r_{t+1}[2]$ is modified by adding t (modulo 2^{32}),
   (b) *Konst* is changed to the output of NLF,
   (c) the output of NLF at $t = 0$ is not used as a keystream word,

where $f16$ is a constant integer $2^{16} + 1 = 65537$.

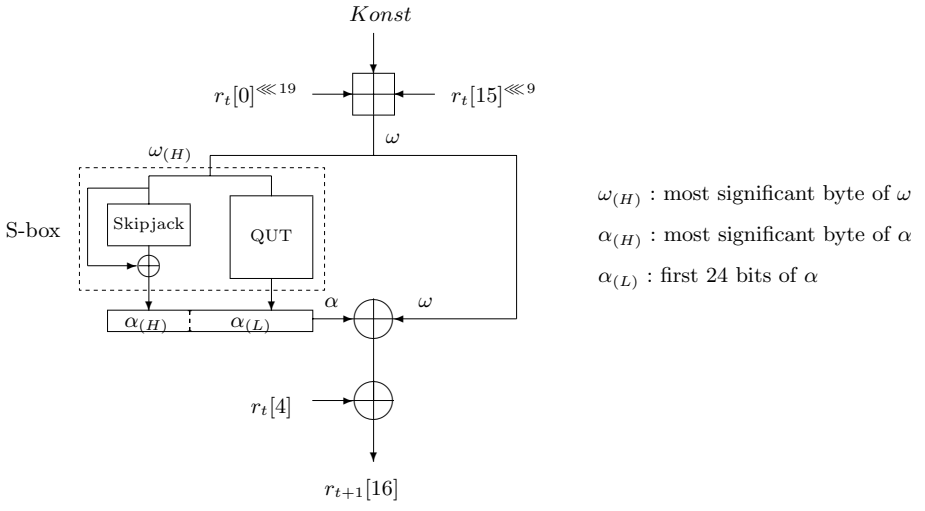


Fig. 1. The update function of NFSR

The f function. The function f is defined as $f(\omega) = \text{S-box}(\omega_{(H)}) \oplus \omega$ where $\omega_{(H)}$ is the most significant 8 bits of 32-bit word ω. The main S-box is composed of two independent smaller S-boxes: the Skipjack S-box (with 8-bit input and 8-bit output) [7] and a custom-designed QUT S-box (with 8-bit input and 24-bit output). The output of main S-box in NLSv2 is defined as a concatenation of outputs of the two smaller S-boxes. Note that the input of Skipjack S-box (that is $\omega_{(H)}$) is added to the output of Skipjack S-box in advance for fast implementation. Since the output of the main S-box is added to ω again, the original output of Skipjack S-box is restored. See Figure 1 for details.

3.2 Non-linear Filter (NLF)

Each output keystream word ν_t of NLF is generated according to the following equation

$$\nu_t = NLF(\sigma_t) = (r_t[0] \boxplus r_t[16]) \oplus (r_t[1] \boxplus r_t[13]) \oplus (r_t[6] \boxplus Konst). \quad (5)$$

Note that there is no output word when $t = 0$ modulo $f16$.

4 Building Linear Approximations

In this section, linear approximations for NLF and NFSR are developed for the CP attack against NLSv2. Our main goal here is to derive new approximations of NFSR that have a higher bias than those presented in [2]. Let n is a positive number. Given a linear approximation $l : \{0,1\}^{2n} \rightarrow \{0,1\}$, a bias ϵ of the approximation l is defined as follows[1]

[1] ϵ is also known in the literature as the correlation or the imbalance.

$$Pr[l = 0] = \frac{1}{2}(1 + \epsilon), \quad |\epsilon| > 0.$$

The advantage of the definition is that the bias of the combination of n independent approximations each of bias ϵ is equal to ϵ^n as asserted by the Piling-up lemma [6].

4.1 Linear Approximations of NFSR

We investigate the bias of the approximation that is a linear combination of two neighboring bits of $\alpha = \text{S-box}(\omega_{(H)})$. As $\omega_{(H)}$ is an 8-bit input, the bias ϵ_i can be calculated as follows

$$\epsilon_i = 2^{-8} \cdot \{\#(\Gamma_i \cdot \alpha = 0) - \#(\Gamma_i \cdot \alpha = 1)\}, \quad i = 0, \ldots, 30.$$

By the exhaustive search, we have found that the linear approximation $\alpha_{29} \oplus \alpha_{30} = 1$ has the largest bias of $2^{-2.3}$. Since $f(\omega) = \text{S-box}(\omega_{(H)}) \oplus \omega$, it is clear that the following output approximation has the bias of $2^{-2.3}$.

$$\Gamma_{29} \cdot (\omega \oplus f(\omega)) = 1 \tag{6}$$

From the structure of the update function of NFSR, we know that the following relation is always true.

$$\Gamma_{29} \cdot (f(\omega)_t \oplus r_t[4] \oplus r_{t+1}[16]) = 0$$

By combining the above relation with Approximation (6), we obtain the approximation

$$\Gamma_{29} \cdot (\omega_t \oplus r_t[4] \oplus r_{t+1}[16]) = 1 \tag{7}$$

that has the bias of $2^{-2.3}$.

4.2 Linear Approximations of NLF

The best linear approximation of NLF for our attack is similar to the one which was given in [2] except that we use the bit position 29 and 30 instead of 12, 13, 22 and 23. The bias of the approximation is given by the following lemma.

Lemma 5. *Given two consecutive outputs of NLF, namely ν_t and ν_{t+1}, the following approximation*

$$\Gamma_i \cdot (\nu_t \oplus \nu_{t+1}) = \Gamma_i \cdot (r_t[0] \oplus r_t[2] \oplus r_t[6] \oplus r_t[7] \oplus r_t[13] \oplus r_t[14] \oplus r_t[16] \oplus r_{t+1}[16])$$

holds with the bias of $\frac{1}{36}(1 + 2^{-2i-1})^2$.

Proof. From Equation (5), we know that

$$\begin{aligned}\nu_t \oplus \nu_{t+1} = {} & (r_t[0] \boxplus r_t[16]) \oplus (\mathbf{r_t[1]} \boxplus r_t[13]) \oplus (r_t[6] \boxplus \mathbf{Konst}) \\ & \oplus (\mathbf{r_{t+1}[0]} \boxplus r_{t+1}[16]) \oplus (r_{t+1}[1] \boxplus r_{t+1}[13]) \oplus (r_{t+1}[6] \boxplus \mathbf{Konst})\end{aligned}$$

for two consecutive clocks $(t, t+1)$. Note that $\mathbf{r_t}[\mathbf{1}]$ and **Konst** are used twice in above expression. Hence, according to Corollary 2, the following two approximations have the probability of $\frac{1}{2}(1+\frac{1}{3}+\frac{1}{3}2^{-2i-1})$ each.

$$\Gamma_i \cdot (r_t[6] \boxplus \mathbf{Konst}) \oplus \Gamma_i \cdot (r_{t+1}[6] \boxplus \mathbf{Konst}) = \Gamma_i \cdot (r_t[6] \oplus r_{t+1}[6])$$
$$\Gamma_i \cdot (\mathbf{r_t}[\mathbf{1}] \boxplus r_t[13]) \oplus \Gamma_i \cdot (\mathbf{r_{t+1}}[\mathbf{0}] \boxplus r_{t+1}[16]) = \Gamma_i \cdot (r_t[13] \oplus r_{t+1}[16])$$

In addition, due to Corollary 1, the approximation given below holds with the probability of $\frac{1}{2}(1+2^{-1})$, respectively.

$$\Gamma_i \cdot (r_t[0] \boxplus r_t[16]) = \Gamma_i \cdot (r_t[0] \oplus r_t[16])$$
$$\Gamma_i \cdot (r_{t+1}[1] \boxplus r_{t+1}[13]) = \Gamma_i \cdot (r_{t+1}[1] \oplus r_{t+1}[13])$$

Hence, the overall bias is $(\frac{1}{3}+\frac{1}{3}2^{-2i-1})^2 \times 2^{-2} = \frac{1}{36}(1+2^{-2i-1})^2$. □

Therefore, the best linear approximation of NLF for our attack is

$$\Gamma_{29} \cdot (\nu_t \oplus \nu_{t+1}) = \Gamma_{29} \cdot (r_t[0] \oplus r_t[2] \oplus r_t[6] \oplus r_t[7] \oplus r_t[13] \oplus r_t[14] \oplus r_t[16] \oplus r_{t+1}[16] \tag{8}$$

that has the bias of $\frac{1}{36}(1+2^{-2\times 29-1})^2 \approx 2^{-5.2}$.

Linear property of NFSR. Due to the update rule of NFSR, we know that $r_{t+i}[j] = r_{t+j}[i]$ where $i, j > 0$.

5 Crossword Puzzle (CP) Attack on NLSv2

In NLSv2, the value of $Konst$ is updated by taking the output of NLF at every 65537 clock. In [2], authors showed that $Konst$ terms could be removed from the distinguisher by combining two consecutive approximations of NLF. In this section, the similar technique is adapted for our attack. That is, the distinguisher are derived by combining the approximations of NFSR and NLF in such a way that the internal states of the shift register are canceled out.

However, we develop more efficient attack on NLSv2 using Approximation (7) and (8) at clock positions η which are

$$\eta = \{0, 2, 6, 7, 13, 14, 16, 17\}.$$

Note that Approximation (7) consists of non-linear terms and linear terms: $\Gamma_{29} \cdot \omega_t$ and $\Gamma_{29} \cdot (r_t[4] \oplus r_{t+1}[16])$, respectively. In the following section, we develop the approximations of the X_t and Y_t separately which are defined as follows:

$$X_t = \bigoplus_{k \in \eta} \Gamma_{29} \cdot (r_{t+k}[4] \oplus r_{t+k+1}[16]), \quad Y_t = \bigoplus_{k \in \eta} \Gamma_{29} \cdot \omega_{t+k}.$$

5.1 Bias of X_t

Due to Approximation (8), the X_t can be represented in the following form:

$$\begin{aligned} X_t &= \bigoplus_{k\in\eta} \Gamma_{29} \cdot (r_{t+k}[4] \oplus r_{t+k+1}[16]) = \bigoplus_{k\in\eta} \Gamma_{29} \cdot (r_{t+4}[k] \oplus r_{t+17}[k]) \\ &= \Gamma_{29} \cdot (\nu_{t+4} \oplus \nu_{t+5} \oplus \nu_{t+17} \oplus \nu_{t+18}). \end{aligned} \tag{9}$$

The bias of Approximation (9) is $2^{-8.6}$. The calculations of the bias are given below. Due to the definition of ν_t given in Equation (5), we know that

$$\begin{aligned} &\Gamma_{29} \cdot (\nu_{t+4} \oplus \nu_{t+5} \oplus \nu_{t+17} \oplus \nu_{t+18}) \\ &= \Gamma_{29} \cdot (r_{t+4}[0] \boxplus r_{t+4}[16]) \oplus \Gamma_{29} \cdot (\mathbf{r_{t+4}[1]} \boxplus \mathbf{r_{t+4}[13]}) \oplus \Gamma_{29} \cdot (r_{t+4}[6] \boxplus \mathbf{Konst}) \\ &\oplus \Gamma_{29} \cdot (\mathbf{r_{t+5}[0]} \boxplus r_{t+5}[16]) \oplus \Gamma_{29} \cdot (r_{t+5}[1] \boxplus \mathbf{r_{t+5}[13]}) \oplus \Gamma_{29} \cdot (r_{t+5}[6] \boxplus \mathbf{Konst}) \\ &\oplus \Gamma_{29} \cdot (\mathbf{r_{t+17}[0]} \boxplus r_{t+17}[16]) \oplus \Gamma_{29} \cdot (\mathbf{r_{t+17}[1]} \boxplus r_{t+17}[13]) \oplus \Gamma_{29} \cdot (r_{t+17}[6] \boxplus \mathbf{Konst}) \\ &\oplus \Gamma_{29} \cdot (\mathbf{r_{t+18}[0]} \boxplus r_{t+18}[16]) \oplus \Gamma_{29} \cdot (r_{t+18}[1] \boxplus r_{t+18}[13]) \oplus \Gamma_{29} \cdot (r_{t+18}[6] \boxplus \mathbf{Konst}) \end{aligned}$$

We can see that several terms are shared due to the linear property of NFSR. Hence, the approximations are applied separately into four groups as follows.

1. According to Corollary 3, we get

$$\begin{aligned} &\Gamma_{29} \cdot (\mathbf{r_{t+4}[1]} \boxplus \mathbf{r_{t+4}[13]}) \oplus \Gamma_{29} \cdot (\mathbf{r_{t+17}[0]} \boxplus r_{t+17}[16]) \oplus \Gamma_{29} \cdot (\mathbf{r_{t+5}[0]} \boxplus r_{t+5}[16]) \\ &= \Gamma_{29} \cdot r_{t+17}[16] \oplus \Gamma_{29} \cdot r_{t+5}[16] \end{aligned}$$

   that holds with the probability of $\frac{29}{48} + \frac{1}{3}2^{-2\times 29-4} \approx \frac{1}{2}(1 + 2^{-2.3})$.
2. Due to Lemma 3, the approximation

$$\begin{aligned} &\Gamma_{29} \cdot (r_{t+5}[1] \boxplus \mathbf{r_{t+5}[13]}) \oplus \Gamma_{29} \cdot (\mathbf{r_{t+18}[0]} \boxplus r_{t+18}[16]) \oplus \Gamma_{29} \cdot (\mathbf{r_{t+17}[1]} \boxplus r_{t+17}[13]) \\ &= \Gamma_{29} \cdot (r_{t+5}[1] \oplus r_{t+5}[13] \oplus r_{t+18}[16] \oplus r_{t+17}[13]) \end{aligned}$$

   holds with the probability of around $\frac{5}{8} = \frac{1}{2}(1 + 2^{-2})$.
3. Lemma 3 also asserts that the approximation

$$\begin{aligned} &\Gamma_{29} \cdot (r_{t+4}[6] \boxplus \mathbf{Konst}) \oplus \Gamma_{29} \cdot (r_{t+5}[6] \boxplus \mathbf{Konst}) \oplus \Gamma_{29} \cdot (r_{t+17}[6] \boxplus \mathbf{Konst}) \\ &\oplus \Gamma_{29} \cdot (r_{t+18}[6] \boxplus \mathbf{Konst}) = \Gamma_{29} \cdot (r_{t+4}[6] \oplus r_{t+5}[6] \oplus r_{t+17}[6] \oplus r_{t+18}[6]) \end{aligned}$$

   holds with the probability of around $\frac{3}{5} = \frac{1}{2}(1 + 2^{-2.3})$.
4. Corollary 1 says that the approximation

$$\begin{aligned} &\Gamma_{29} \cdot (r_{t+4}[0] \boxplus r_{t+4}[16]) \oplus \Gamma_{29} \cdot (r_{t+18}[1] \boxplus r_{t+18}[13]) \\ &= \Gamma_{29} \cdot (r_{t+4}[0] \oplus r_{t+4}[16]) \oplus \Gamma_{29} \cdot (r_{t+18}[1] \oplus r_{t+18}[13]) \end{aligned}$$

   holds with the probability of $\frac{1}{2}(1 + 2^{-2})$.

Therefore, the bias of Approximation (9) is $2^{-2.3} \times 2^{-2} \times 2^{-2.3} \times 2^{-2} = 2^{-8.6}$.

5.2 Bias of Y_t

The ω_t is an intermediate variable that is defined as $\omega_t = (r_t[0]^{\lll 19}) \boxplus (r_t[15]^{\lll 9}) \boxplus Konst$. Due to Lemma 2, the ω_t has the following approximation

$$\begin{aligned} \Gamma_{29} \cdot \omega &= \Gamma_{29} \cdot (r_t[0]^{\lll 19} \oplus r_t[15]^{\lll 9} \oplus Konst) \\ &= \Gamma_{10} \cdot r_t[0] \oplus \Gamma_{20} \cdot r_t[15] \oplus \Gamma_{29} \cdot Konst \end{aligned}$$

that holds with the probability of $\frac{2}{3} - \frac{1}{3}2^{-2\times 29-1} \approx \frac{1}{2}(1+2^{-1.6})$. Due to Lemma 5, the approximation of Y_t can be described as

$$\begin{aligned} Y_t &= \bigoplus_{k\in\eta} \Gamma_{29} \cdot \omega_{t+k} = \bigoplus_{k\in\eta} (\Gamma_{10} \cdot r_{t+k}[0] \oplus \Gamma_{20} \cdot r_{t+k}[15] \oplus \Gamma_{29} \cdot Konst) \\ &= \Gamma_{10} \cdot (\nu_t \oplus \nu_{t+1}) \oplus \Gamma_{20} \cdot (\nu_{t+15} \oplus \nu_{t+16}). \end{aligned} \tag{10}$$

If Lemma 2 and Lemma 5 are independently applied to the Approximation (10), the bias is expected to be $2^{-1.6*8-5.2} = 2^{-18.0}$. However, due to the dependencies of approximations, the bias of Approximation (10) is surprisingly around $2^{-10.4}$. The detail analysis on the bias will be discussed in Section 5.4.

We note that $Konst$ terms have disappeared since the binary addition of eight approximations cancels $Konst$ as observed in [2]. Due to the lack of a keystream word at every f16-th clock, we can see precisely when $Konst$ is updated. Since the updated $Konst$ has been effective to all states of registers after the first 17 clocks, the observations generated from the first 17 clocks should not be counted for the bias. Hence, $Konst$ is regarded as a constant in all approximations.[2]

5.3 Bias of the Distinguisher

From Approximation (7),

$$\bigoplus_{k\in\eta} \Gamma_{29} \cdot (\omega_{t+k} \oplus r_{t+k}[4] \oplus r_{t+1+k}[16]) = X_t \oplus Y_t = 0 \tag{11}$$

On the other hand, by adding up the approximations of (9) and (10), we obtain the following approximation

$$X_t \oplus Y_t = \Gamma_{29} \cdot (\nu_{t+4} \oplus \nu_{t+5} \oplus \nu_{t+17} \oplus \nu_{t+18}) \oplus \Gamma_{10} \cdot (\nu_t \oplus \nu_{t+1}) \oplus \Gamma_{20} \cdot (\nu_{t+15} \oplus \nu_{t+16}) \tag{12}$$

that holds with the bias equal to $2^{-8.6} \times 2^{-10.4}$. Therefore, by combining (11) and (12), the distinguisher on NLSv2 can be described by the approximation

$$\Gamma_{29} \cdot (\nu_{t+4} \oplus \nu_{t+5} \oplus \nu_{t+17} \oplus \nu_{t+18}) \oplus \Gamma_{10} \cdot (\nu_t \oplus \nu_{t+1}) \oplus \Gamma_{20} \cdot (\nu_{t+15} \oplus \nu_{t+16}) = 0 \tag{13}$$

that holds with the bias of around $2^{-2.3\times 8} \times 2^{-8.6} \times 2^{-10.4} = 2^{-37.4}$.

[2] By this reason, the notation $Konst_t$ is not used in the approximations.

5.4 The Bias of Approximation (10)

According to the definition of ν_t given by Equation (5), we can write the following approximation

$$
\begin{aligned}
&\Gamma_{10} \cdot (\nu_t \oplus \nu_{t+1}) \oplus \Gamma_{20} \cdot (\nu_{t+15} \oplus \nu_{t+16}) \\
&= \Gamma_{10} \cdot (r_t[0] \boxplus r_t[16]) \oplus \Gamma_{10} \cdot (r_t[1] \boxplus r_t[13]) \Gamma_{10} \cdot (r_t[6] \boxplus Konst) \\
&\oplus \Gamma_{10} \cdot (r_{t+1}[0] \boxplus r_{t+1}[16]) \oplus \Gamma_{10} \cdot (r_{t+1}[1] \boxplus r_{t+1}[13]) \oplus \Gamma_{10} \cdot (r_{t+1}[6] \boxplus Konst) \\
&\oplus \Gamma_{20} \cdot (r_{t+15}[0] \boxplus r_{t+15}[16]) \oplus \Gamma_{20} \cdot (r_{t+15}[1] \boxplus r_{t+15}[13]) \oplus \Gamma_{20} \cdot (r_{t+15}[6] \boxplus Konst) \\
&\oplus \Gamma_{20} \cdot (r_{t+16}[0] \boxplus r_{t+16}[16]) \oplus \Gamma_{20} \cdot (r_{t+16}[1] \boxplus r_{t+16}[13]) \oplus \Gamma_{20} \cdot (r_{t+16}[6] \boxplus Konst) \\
&\triangleq \Delta_1 \oplus \Delta_2 \oplus \Delta_3
\end{aligned}
$$

where

$$
\begin{aligned}
\Delta_1 &= \Gamma_{10} \cdot (r_t[0] \boxplus r_t[16]) \oplus \Gamma_{20} \cdot (r_{t+15}[0] \boxplus r_{t+15}[16]) \\
&\quad \oplus \Gamma_{10} \cdot (r_{t+1}[1] \boxplus r_{t+1}[13]) \oplus \Gamma_{20} \cdot (r_{t+16}[1] \boxplus r_{t+16}[13]) \\
\Delta_2 &= \Gamma_{10} \cdot (r_t[1] \boxplus r_t[13]) \oplus \Gamma_{20} \cdot (r_{t+15}[1] \boxplus r_{t+15}[13]) \\
&\quad \oplus \Gamma_{10} \cdot (r_{t+1}[0] \boxplus r_{t+1}[16]) \oplus \Gamma_{20} \cdot (r_{t+16}[0] \boxplus r_{t+16}[16]) \\
\Delta_3 &= \Gamma_{10} \cdot (r_t[6] \boxplus Konst) \oplus \Gamma_{20} \cdot (r_{t+15}[6] \boxplus Konst) \\
&\quad \oplus \Gamma_{10} \cdot (r_{t+1}[6] \boxplus Konst) \oplus \Gamma_{20} \cdot (r_{t+16}[6] \boxplus Konst)
\end{aligned}
$$

In order to determine the bias of Δ_1, Δ_2 and Δ_3, the following two lemmas are required.

Lemma 6. *Given $x, y, a, b, c, d, k \in \{0,1\}^{32}$, the following approximation has the bias of $2^{-3.1}$ when $i > 0$.*

$$
\begin{aligned}
&\Gamma_i \cdot (x \boxplus a) \oplus \Gamma_i \cdot (y \boxplus b) \oplus \Gamma_i \cdot (x \boxplus c) \oplus \Gamma_i \cdot (y \boxplus d) \\
&= \Gamma_i \cdot (a \boxplus b \boxplus k) \oplus \Gamma_i \cdot (c \boxplus d \boxplus k)
\end{aligned}
$$

Proof. For the proof, see Appendix F.

Lemma 7. *Given $x, y, z, w, a, b, c, d, k \in \{0,1\}^{32}$, the following approximation holds with the bias of $2^{-4.2}$ when $i > 0$.*

$$
\begin{aligned}
&\Gamma_i \cdot (x \boxplus a) \oplus \Gamma_i \cdot (y \boxplus b) \oplus \Gamma_i \cdot (z \boxplus c) \oplus \Gamma_i \cdot (w \boxplus d) \\
&= \Gamma_i \cdot (x \boxplus y \boxplus k) \oplus \Gamma_i \cdot (a \boxplus b \boxplus k) \oplus \Gamma_i \cdot (z \boxplus w \boxplus k) \oplus \Gamma_i \cdot (c \boxplus d \boxplus k) \quad (14)
\end{aligned}
$$

Proof. See Appendix G for the proof.

Now we can derive the biases of the approximations Δ_1, Δ_2 and Δ_3.

Δ_1: From the definition of the rotations, we know that

$$
\begin{aligned}
\Delta_1 &= \Gamma_{29} \cdot (r_t[0]^{\lll 19} \boxplus r_t[16]^{\lll 19}) \oplus \Gamma_{29} \cdot (r_{t+15}[0]^{\lll 9} \boxplus r_{t+15}[16]^{\lll 9}) \\
&\quad \oplus \Gamma_{29} \cdot (r_{t+1}[1]^{\lll 19} \boxplus r_{t+1}[13]^{\lll 19}) \oplus \Gamma_{29} \cdot (r_{t+16}[1]^{\lll 9} \boxplus r_{t+16}[13]^{\lll 9})
\end{aligned}
$$

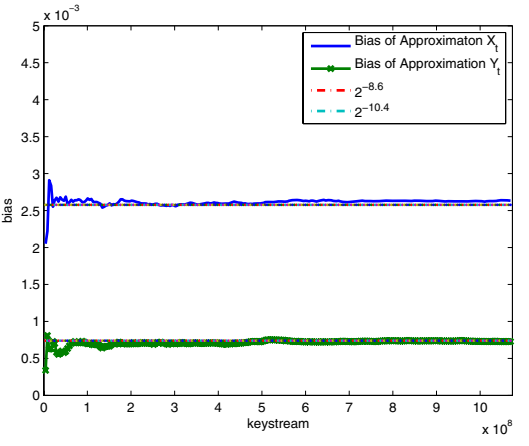


Fig. 2. The biases of Approximation (9) and (10)

According to Lemma 7, the following approximation holds with the bias of $2^{-4.2}$.

$$\begin{aligned}\Delta_1 &= \Gamma_{29}\cdot(r_t[0]^{\lll 19} \boxplus r_{t+15}[0]^{\lll 9} \boxplus Konst) \oplus \Gamma_{29}\cdot(r_t[16]^{\lll 19} \boxplus r_{t+15}[16]^{\lll 9} \boxplus Konst)\\ &\quad \oplus\Gamma_{29}\cdot(r_{t+1}[1]^{\lll 19} \boxplus r_{t+16}[1]^{\lll 9} \boxplus Konst) \oplus \Gamma_{29}\cdot(r_{t+1}[13]^{\lll 19} \boxplus r_{t+16}[13]^{\lll 9} \boxplus Konst)\\ &= \Gamma_{29}\cdot(\omega_t \oplus \omega_{t+16} \oplus \omega_{t+2} \oplus \omega_{t+14})\end{aligned}$$

Δ_2 and Δ_3: Due to Lemma 6, we can write the approximations

$$\begin{aligned}\Delta_2 &= \Gamma_{29}\cdot(r_t[1]^{\lll 19} \boxplus r_{t+15}[1]^{\lll 9}) \oplus \Gamma_{29}\cdot(r_t[13]^{\lll 19} \boxplus r_{t+15}[13]^{\lll 9})\\ &\quad \oplus\Gamma_{29}\cdot(r_{t+1}[0]^{\lll 19} \boxplus r_{t+16}[0]^{\lll 9}) \oplus \Gamma_{29}\cdot(r_{t+1}[16]^{\lll 19} \boxplus r_{t+16}[16]^{\lll 9})\\ &= \Gamma_{29}\cdot(r_t[13]^{\lll 19} \boxplus r_{t+15}[13]^{\lll 9} \boxplus Konst) \oplus \Gamma_{29}\cdot(r_{t+1}[16]^{\lll 19} \boxplus r_{t+16}[16]^{\lll 9} \boxplus Konst)\\ &= \Gamma_{29}\cdot(\omega_{t+13} \oplus \omega_{t+17})\\ \Delta_3 &= \Gamma_{29}\cdot(r_t[6]^{\lll 19} \boxplus r_{t+15}[6]^{\lll 9}) \oplus \Gamma_{29}\cdot(Konst^{\lll 19} \boxplus Konst^{\lll 9})\\ &\quad \oplus\Gamma_{29}\cdot(r_{t+1}[6]^{\lll 19} \boxplus r_{t+16}[6]^{\lll 9}) \oplus \Gamma_{29}\cdot(Konst^{\lll 19} \boxplus Konst^{\lll 9})\\ &= \Gamma_{29}\cdot(r_t[6]^{\lll 19} \boxplus r_{t+15}[6]^{\lll 9} \boxplus Konst) \oplus \Gamma_{29}\cdot(r_{t+1}[6]^{\lll 19} \boxplus r_{t+16}[6]^{\lll 9} \boxplus Konst)\\ &= \Gamma_{29}\cdot(\omega_{t+6} \oplus \omega_{t+7})\end{aligned}$$

with the same bias of $2^{-3.1}$. Thus, Approximation (10) holds with the bias of $2^{-(4.2+3.1\times 2)} = 2^{-10.4}$.

5.5 Experiments

The verification of the bias of Distinguisher (13) is not feasible due to the requirement of large observations of keystream. Instead, our experiments have been focused on verifying the biases of Approximation (9) and (10) independently. Figure 2 verifies that the estimated biases of those approximations are correct.

6 Conclusion

In this paper, we present a Crossword Puzzle (CP) attack against NLSv2 that is a tweaked version of NLS. Even though the designers of NLSv2 aimed to avoid the distinguishing attack that was constructed for the NLS, we have shown that the CP attack can be applied for NLSv2. The distinguisher has a bias higher than 2^{-40} and consequently, the attack requires less than 2^{80} observations which was given as the security benchmark by the designers.

Acknowledgment. We are grateful to anonymous referees of ISC 2007 for their invaluable comments. The authors were supported by ARC grants DP0451484, DP0663452 and 2006 MQSN grant.

References

1. Cho, J.Y., Pieprzyk, J.: Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 49–64. Springer, Heidelberg (2004)
2. Cho, J.Y., Pieprzyk, J.: Crossword puzzle attack on NLS. In: Proceedings of Selected Areas in Cryptography - SAC 2006, Montreal, Quebec, Canada (August 2006)
3. Cho, J.Y., Pieprzyk, J.: Linear distinguishing attack on NLS. In: SASC, workshop, (2006), Available at http://www.ecrypt.eu.org/stvl/sasc2006/
4. Hawkes, P., Paddon, M., Rose, G., de Vries, M.W.: Primitive specification for NLS(April 2005), Available at http://www.ecrypt.eu.org/stream/nls.html
5. Hawkes, P., Paddon, M., Rose, G., de Vries, M.W.: Primitive specification for NLSv2. eSTREAM (March 2006), Available at http://www.ecrypt.eu.org/stream/nls.html
6. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
7. NIST. SKIPJACK and KEA algorithm specifications (May 1998), Available at http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack.pdf
8. ECRYPT NoE. eSTREAM - the ECRYPT stream cipher project (2005), Available at http://www.ecrypt.eu.org/stream/
9. Nyberg, K., Wallen, J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006)

A Proof of Lemma 2

By Definition (1), we obtain

$$\Gamma_i \cdot (x \boxplus y \boxplus z) = \Gamma_i \cdot (x \oplus y \oplus z) \oplus \Gamma_{i-1} \cdot (R(x,y) \oplus R(x \boxplus y, z)).$$

Thus, our task is to find $Pr[\Gamma_{i-1} \cdot (R(x,y) \oplus R(x \boxplus y, z)) = 0]$. Let us denote $L_i = x_{(i)} \oplus y_{(i)} \oplus z_{(i)}$, $Q_i = x_{(i)}y_{(i)} \oplus y_{(i)}z_{(i)} \oplus z_{(i)}x_{(i)}$, and $T_i = x_{(i)}y_{(i)}z_{(i)}$. Assume further that X_i and Y_i are defined as follows.

$$X_i \triangleq R(x,y)_{(i)} \oplus R(x \boxplus y, z)_{(i)} = Q_i \oplus L_i X_{i-1} \oplus Y_{i-1}$$
$$Y_i \triangleq R(x,y)_{(i)} R(x \boxplus y, z)_{(i)} = T_i X_{i-1} \oplus Q_i Y_{i-1}$$

Since $Q_i \cdot L_i = T_i$ by definition, the following relation between X_i and Y_i holds

$$Y_i = Q_i X_i \oplus Q_i.$$

We try to find out the $Pr[X_i = 0]$. We start from the equation $X_i = Q_i \oplus L_i X_{i-1} \oplus Y_{i-1}$ and replace Y_{i-1} by $Y_{i-1} = Q_{i-1} X_i \oplus Q_{i-1}$, so we find

$$X_i = Q_i \oplus L_i X_{i-1} \oplus Y_{i-1} = Q_i \oplus Q_{i-1} \oplus (L_i \oplus Q_{i-1}) X_{i-1}. \tag{15}$$

This gives us

$$Pr[X_i = 0] = \frac{1}{2} Pr[X_{i-1} = 0] + \frac{1}{4}(1 - Pr[X_{i-1} = 0]) = \frac{1}{4} + \frac{1}{4} Pr[X_{i-1} = 0]$$

Therefore, applying the recursion relation from Appendix H, we obtain

$$Pr[X_i = 0] = \frac{1}{3} + \frac{1}{3} 2^{-2i-1}. \tag{16}$$

Note that $Pr[X_0 = 0] = Pr[x_{(0)} y_{(0)} \oplus y_{(0)} z_{(0)} \oplus z_{(0)} x_{(0)} = 0] = \frac{1}{2}$. Hence, we can write that

$$\begin{aligned} \Gamma_{i-1} \cdot (R(x, y) \oplus R(x \boxplus y, z)) &= X_{i-1} \oplus X_i = Q_i \oplus (L_i \oplus 1) X_{i-1} \oplus Y_{i-1} \\ &= Q_i \oplus Q_{i-1} \oplus (L_i \oplus Q_{i-1} \oplus 1) X_{i-1} \end{aligned}$$

Therefore,

$$Pr[\Gamma_{i-1} \cdot (R(x, y) \oplus R(x \boxplus y, z)) = 0] = \begin{cases} Pr[Q_i \oplus Q_{i-1} = 0] = \frac{1}{2}, & \text{if } X_{i-1} = 0, \\ Pr[Q_i \oplus L_i \oplus 1 = 0] = \frac{3}{4}, & \text{if } X_{i-1} = 1 \end{cases}$$

By applying Equation (16), we get the final result

$$Pr[\Gamma_{i-1} \cdot (R(x, y) \oplus R(x \boxplus y, z))] = \frac{1}{2} Pr[X_{i-1} = 0] + \frac{3}{4}(1 - Pr[X_{i-1} = 0]) = \frac{2}{3} - \frac{1}{3} 2^{-2i-1}$$

B Proof of Lemma 3

Let us denote $\Phi_{n,(i)} = R(x_1, k)_{(i)} \oplus R(x_2, k)_{(i)} \oplus \cdots \oplus R(x_n, k)_{(i)}$. By Relation (2), we know

$$\begin{aligned} \Phi_{n,(i)} = {} & k_{(i)}(x_{1,(i)} \oplus x_{2,(i)} \oplus \cdots \oplus x_{n,(i)}) \oplus (x_{1,(i)} \oplus k_{(i)}) R(x_1, k)_{(i-1)} \oplus \\ & (x_{2,(i)} \oplus k_{(i)}) R(x_2, k)_{(i-1)} \oplus \cdots \oplus (x_{n,(i)} \oplus k_{(i)}) R(x_n, k)_{(i-1)} \end{aligned}$$

Then, $\Phi_{n,(i)}$ has the following properties.

- If $\bigoplus_{t=1}^{n} x_{t,(i)} = 0$, then there exists a pair of $(x_{1,(i)}, x_{2,(i)}, \ldots, x_{n,(i)}, k_{(i)})$ which generate the same $\Phi_{n,(i)}$.

– If $\bigoplus_{t=1}^{n} x_{t,(i)} = 1$, then there exists a pair of $(x_{1,(i)}, x_{2,(i)}, \ldots, x_{n,(i)}, k_{(i)})$ whose $\Phi_{n,(i)}$s are complement each other.

Hence, by defining, $P_{r,(i)} = Pr[\bigoplus_{t=1}^{r} R(x_t, k)_{(i)} = 0]$, we get

$$P_{n,(i)} = \frac{1}{2^{n+1}}[\sum_{r=0}^{n/2} \binom{n}{2r} 2P_{2r,(i-1)} + \sum_{r=0}^{n/2-1} \binom{n}{2r+1}] = \frac{1}{4} + \frac{1}{2^n} \sum_{r=0}^{n/2} \binom{n}{2r} P_{2r,(i-1)}$$

where $P_0 = 1$. Hence, $P_{n,(i)} \approx \frac{n+2}{2(n+1)}$ for $i > 0$.

By definition, we can write $(x \boxplus k)_{(i)} = x_{(i)} \oplus k_{(i)} \oplus R(x,k)_{(i-1)}$. Thus, we get

$$\begin{aligned}
&\Gamma_i \cdot (x_1 \boxplus k) \oplus \Gamma_i \cdot (x_2 \boxplus k) \oplus \cdots \oplus \Gamma_i \cdot (x_n \boxplus k) \oplus \Gamma_i \cdot (x_1 \oplus x_2 \oplus \cdots \oplus x_n) \\
&= \Gamma_{i-1} \cdot (R(x_1,k) \oplus R(x_2,k) \oplus \cdots \oplus R(x_n,k)) \\
&= \Phi_{n,(i-1)} \oplus \Phi_{n,(i)} \\
&= k_{(i)}(x_{1,(i)} \oplus x_{2,(i)} \oplus \cdots \oplus x_{n,(i)}) \oplus (x_{1,(i)} \oplus k_{(i)} \oplus 1)R(x_1,k)_{(i-1)} \oplus \\
&(x_{2,(i)} \oplus k_{(i)} \oplus 1)R(x_2,k)_{(i-1)} \oplus \cdots \oplus (x_{n,(i)} \oplus k_{(i)} \oplus 1)R(x_n,k)_{(i-1)}
\end{aligned}$$

As before, we can get the following equation

$$Pr[\Phi_{n,(i-1)} \oplus \Phi_{n,(i)} = 0] = \frac{1}{4} + \frac{1}{2^n} \sum_{r=0}^{n/2} \binom{n}{2r} P_{n-2r,(i-1)}$$

$$= \frac{1}{4} + \frac{1}{2^n} \sum_{r=0}^{n/2} \binom{n}{n-2r} P_{n-2r,(i-1)} = P_{n,(i)}$$

For $i > 0$, we have $Pr[\Phi_{n,(i-1)} \oplus \Phi_{n,(i)} = 0] \approx \frac{n+2}{2(n+1)}$ which concludes the proof.

C Proof of Corollary 2

From Definition (1), we write

$$R(x,y)_{(i)} \oplus R(x,z)_{(i)} = x_{(i)}y_{(i)} \oplus (x_{(i)} \oplus y_{(i)})R(x,y)_{(i-1)} \oplus x_{(i)}z_{(i)} \oplus (x_{(i)} \oplus z_{(i)})R(x,z)_{(i-1)}.$$

Then, according to $(x_{(i)}, y_{(i)}, z_{(i)})$, the expression $R(x,y)_{(i)} \oplus R(x,z)_{(i)}$ is split into eight cases. Hence, we have the following recursive probability

$$Pr[R(x,y)_{(i)} \oplus R(x,z)_{(i)} = 0] = \frac{1}{2} + \frac{1}{4} Pr[R(x,y)_{(i-1)} \oplus R(x,z)_{(i-1)} = 0].$$

Using the recursion relation from Appendix H, we state that

$$Pr[R(x,y)_{(i)} \oplus R(x,z)_{(i)} = 0] = \frac{2}{3} + \frac{1}{3} 2^{-2i-2}$$

Applying Relation (2), we can get

$$\Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (x \boxplus z) \oplus \Gamma_i \cdot (y \oplus z) = \Gamma_{i-1} \cdot (R(x,y) \oplus R(x,z))$$
$$= x_{(i)}y_{(i)} \oplus (x_{(i)} \oplus y_{(i)} \oplus 1)R(x,y)_{(i-1)} \oplus x_{(i)}z_{(i)} \oplus (x_{(i)} \oplus z_{(i)} \oplus 1)R(x,z)_{(i-1)}$$

Therefore, arguing in similar way as above, we establish that

$$Pr[\Gamma_i \cdot (R(x,y) \oplus R(x,z)) = 0] = \frac{1}{2} + \frac{1}{4}Pr[R(x,y)_{(i-1)} \oplus R(x,z)_{(i-1)} = 0]$$
$$= \frac{2}{3} + \frac{1}{3}2^{-2i-2}.$$

D Proof of Lemma 4

Our task is to determine the probability of the following approximation:

$$\Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (z \boxplus w) = \Gamma_i \cdot (x \boxplus z) \oplus \Gamma_i \cdot (y \boxplus w).$$

We add both sides of the approximation and are going to find the probability that it becomes zero. So we have

$$\Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (z \boxplus w) \oplus \Gamma_i \cdot (x \boxplus z) \oplus \Gamma_i \cdot (y \boxplus w)$$
$$= \Gamma_{i-1} \cdot (R(x,y) \oplus R(z,w) \oplus R(x,z) \oplus R(y,w))$$
$$= x_{(i)}y_{(i)} \oplus z_{(i)}w_{(i)} \oplus x_{(i)}z_{(i)} \oplus y_{(i)}w_{(i)} \oplus (x_{(i)} \oplus y_{(i)} \oplus 1)R(x,y)_{(i-1)}$$
$$\oplus (z_{(i)} \oplus w_{(i)} \oplus 1)R(z,w)_{(i-1)} \oplus (x_{(i)} \oplus z_{(i)} \oplus 1)R(x,z)_{(i-1)} \oplus (y_{(i)} \oplus w_{(i)} \oplus 1)R(y,w)_{(i-1)}$$
$$\triangleq \Lambda_i$$

Then Λ_i can be split into eight cases according to the values of $(x_{(i)}, y_{(i)}, z_{(i)}, w_{(i)})$. In order to compute $Pr[\Lambda_i = 0]$, the following three probabilities are required.

- $\alpha_i = Pr[R(x,y)_{(i)} \oplus R(z,w)_{(i)} \oplus 1 = 0]$,
- $\beta_i = Pr[R(x,y)_{(i)} \oplus R(x,z)_{(i)} = 0]$,
- $\gamma_i = Pr[R(x,y)_{(i)} \oplus R(z,w)_{(i)} \oplus R(x,z)_{(i)} \oplus R(y,w)_{(i)} = 0]$.

They can be used to state that

$$Pr[\Lambda_i = 0] = \frac{1}{4}\alpha_{i-1} + \frac{1}{2}\beta_{i-1} + \frac{1}{8}\gamma_{i-1} + \frac{1}{8} \tag{17}$$

Now the probabilities α_i, β_i and γ_i are computed as follows.

(1) From Lemma 1, we get $\alpha_i = \frac{3}{8} + \frac{1}{4}\alpha_{i-1}$. Hence, $\alpha_i = \frac{1}{2} - 2^{-2i-3}$ by Appendix H.
(2) Using Appendix C, we get $\beta_i = \frac{1}{2} + \frac{1}{4}\beta_{i-1}$. Hence, $\beta_i = \frac{2}{3} + \frac{1}{3}2^{-2i-2}$.
(3) By definition, we see that

$$R(x,y)_{(i)} \oplus R(z,w)_{(i)} \oplus R(x,z)_{(i)} \oplus R(y,w)_{(i)}$$
$$= x_{(i)}y_{(i)} \oplus z_{(i)}w_{(i)} \oplus x_{(i)}z_{(i)} \oplus y_{(i)}w_{(i)} \oplus (x_{(i)} \oplus y_{(i)})R(x,y)_{(i-1)}$$
$$\oplus (z_{(i)} \oplus w_{(i)})R(z,w)_{(i-1)} \oplus (x_{(i)} \oplus z_{(i)})R(x,z)_{(i-1)} \oplus (y_{(i)} \oplus w_{(i)})R(y,w)_{(i-1)}$$

According to the values of $(x_{(i)}, y_{(i)}, z_{(i)}, w_{(i)})$, we establish that

$$\begin{aligned}\gamma_i &= \frac{1}{4}\alpha_{i-1} + \frac{1}{2}\beta_{i-1} + \frac{1}{8}\gamma_{i-1} + \frac{1}{8} \\ &= \frac{1}{4}\sum_{j=0}^{i-1}\alpha_j 2^{-3(i-j-1)} + \frac{1}{2}\sum_{j=0}^{i-1}\beta_j 2^{-3(i-j-1)} + 2^{-3i}\gamma_0 + \frac{1}{7}(1-2^{-3i}) \\ &= \frac{2}{3} + \frac{1}{3}2^{-2i-2}\end{aligned}$$

Therefore, by plugging in the Equation (17), the probability becomes

$$Pr[\Lambda_i = 0] = \frac{1}{4}(\frac{1}{2} - 2^{-2i-1}) + \frac{1}{2}(\frac{2}{3} + \frac{1}{3}2^{-2i}) + \frac{1}{8}(\frac{2}{3} + \frac{1}{3}2^{-2i}) + \frac{1}{8} = \frac{2}{3} + \frac{1}{3}2^{-2i-2}$$

and gives the final result.

E Proof of Corollary 3

We take both sides of the approximation, add them and find the probability when it becomes zero so

$$\begin{aligned}&\Gamma_i{\cdot}(x \boxplus y) \oplus \Gamma_i{\cdot}(x \boxplus z) \oplus \Gamma_i{\cdot}(y \boxplus w) \oplus \Gamma_i{\cdot}(z \oplus w) \\ &= \Gamma_{i-1}{\cdot}(R(x,y) \oplus R(x,z) \oplus R(y,w)) \\ &= x_{(i)}y_{(i)} \oplus (x_{(i)} \oplus y_{(i)} \oplus 1)R(x,y)_{(i-1)} \oplus x_{(i)}z_{(i)} \oplus (x_{(i)} \oplus z_{(i)} \oplus 1)R(x,z)_{(i-1)} \\ &\oplus y_{(i)}w_{(i)} \oplus (y_{(i)} \oplus w_{(i)} \oplus 1)R(y,w)_{(i-1)}\end{aligned}$$

Next, the expression $\Gamma_i{\cdot}(R(x,y) \oplus R(x,z) \oplus R(y,w))$ is split into the sixteen cases according to $(x_{(i)}, y_{(i)}, z_{(i)}, w_{(i)})$. Note that there are four pairs which are complement of each other. Using the notation of Appendix D, we get

$$\alpha_i = Pr[1 \oplus R(x,z)_i \oplus R(y,w)_i = 0] = \frac{1}{2} - 2^{-2i-3}$$

$$\beta_i = Pr[R(x,y)_i \oplus R(x,z)_i = 0] = Pr[R(x,y)_i \oplus R(y,w)_i = 0] = \frac{2}{3} + \frac{1}{3}2^{-2i-2}$$

Therefore, we get the final result

$$\begin{aligned}&Pr[\Gamma_{i-1}{\cdot}(R(x,y) \oplus R(x,z) \oplus R(y,w)) = 0] = \frac{3}{8} + \frac{1}{4}\beta_{(i-1)} + \frac{1}{16}\alpha_{(i-1)} \\ &= \frac{3}{8} + \frac{1}{4}(\frac{2}{3} + \frac{1}{3}2^{-2i}) + \frac{1}{8}(\frac{1}{2} - 2^{-2i-1}) = \frac{29}{48} + \frac{1}{3}2^{-2i-4}\end{aligned}$$

F Proof of Lemma 6

From the approximation being considered, w.l.g we assume that $x = 0$ and $y = 0$ since the variables x and y are independent on the expressions $(a \boxplus b \boxplus k)$ and $(c \boxplus d \boxplus k)$. Then, the approximation is simplified as follows.

$$\Gamma_i\cdot(x \boxplus a) \oplus \Gamma_i\cdot(y \boxplus b) \oplus \Gamma_i \cdot (x \boxplus c) \oplus \Gamma_i\cdot(y \boxplus d) \oplus \Gamma_i\cdot(a \boxplus b \boxplus k) \oplus \Gamma_i\cdot(c \boxplus d \boxplus k)$$
$$= \Gamma_{i-1} \cdot (R(a,b) \oplus R(a \boxplus b, k)) \oplus \Gamma_{i-1} \cdot (R(c,d) \oplus R(c \boxplus d, k))$$

Using the recursive relation (15) in Appendix A, we have

$$(R(a,b) \oplus R(a \boxplus b,k))_{(i)} \oplus (R(c,d) \oplus R(c \boxplus d,k))_{(i)}$$
$$= Q_{1,(i)} \oplus Q_{1,(i-1)} \oplus (L_{1,(i)} \oplus Q_{1,(i-1)})(R(a,b)_{(i-1)} \oplus R(a \boxplus b,k)_{(i-1)}) \oplus$$
$$Q_{2,(i)} \oplus Q_{2,(i-1)} \oplus (L_{2,(i)} \oplus Q_{2,(i-1)})(R(c,d)_{(i-1)} \oplus R(c \boxplus d,k)_{(i-1)})$$

where $Q_{1,(i)} = a_{(i)}b_{(i)} \oplus b_{(i)}k_{(i)} \oplus k_{(i)}a_{(i)}$, $Q_{2,(i)} = c_{(i)}d_{(i)} \oplus d_{(i)}k_{(i)} \oplus k_{(i)}c_{(i)}$, $L_{1,(i)} = a_{(i)} \oplus b_{(i)} \oplus k_{(i)}$ and $L_{2,(i)} = c_{(i)} \oplus d_{(i)} \oplus k_{(i)}$. According to the values of ten variables $(a_{(i)}, b_{(i)}, c_{(i)}, d_{(i)}, k_{(i)}, a_{(i-1)}, b_{(i-1)}, c_{(i-1)}, d_{(i-1)}, k_{(i-1)})$, the above expression is simplified as a function of $(R(a,b)_{(i-1)} \oplus R(a \boxplus b,k)_{(i-1)})$ and $(R(c,d)_{(i-1)} \oplus R(c \boxplus d,k)_{(i-1)})$. Hence, by counting appropriate probabilities, we get

$$Pr[(R(a,b) \oplus R(a \boxplus b,k))_{(i)} \oplus (R(c,d) \oplus R(c \boxplus d,k))_{(i)} = 0]$$
$$= \frac{35}{64} - \frac{3}{64} \cdot Pr[(R(a,b) \oplus R(a \boxplus b,k))_{(i-1)} = 0] - \frac{3}{64} \cdot Pr[(R(c,d) \oplus R(c \boxplus d,k))_{(i-1)} = 0]$$
$$+\frac{5}{64} \cdot Pr[(R(a,b) \oplus R(a \boxplus b,k))_{(i-1)} \oplus (R(c,d) \oplus R(c \boxplus d,k))_{(i-1)} = 0]$$

From Lemma 2, we know that

$$Pr[(R(a,b) \oplus R(a \boxplus b,k))_{(i-1)} = 0] = Pr[(R(c,d) \oplus R(c \boxplus d,k))_{(i-1)} = 0] = \frac{1}{3} + \frac{1}{3}2^{-2i+1}$$

Therefore, by the recursive relation of Appendix H, for $i > 0$,

$$Pr[(R(a,b) \oplus R(a \boxplus b,k))_{(i)} \oplus (R(c,d) \oplus R(c \boxplus d,k))_{(i)} = 0] \approx \frac{33}{59} = \frac{1}{2}(1 + 2^{-3.1})$$

Since $Pr[(R(a,b) \oplus R(a \boxplus b,k))_{(i)} \oplus (R(c,d) \oplus R(c \boxplus d,k))_{(i)} = 0]$ is identical to $Pr[\Gamma_{i-1} \cdot (R(a,b) \oplus R(a \boxplus b,k)) \oplus \Gamma_{i-1} \cdot (R(c,d) \oplus R(c \boxplus d,k)) = 0]$, the lemma holds.

G Proof of Lemma 7

Suppose $k = 0$. Then, the approximation (14) is divided into two independent approximations as follows.

$$\Gamma_i \cdot (x \boxplus a) \oplus \Gamma_i \cdot (y \boxplus b) = \Gamma_i \cdot (x \boxplus y) \oplus \Gamma_i \cdot (a \boxplus b)$$
$$\Gamma_i \cdot (z \boxplus c) \oplus \Gamma_i \cdot (w \boxplus d) = \Gamma_i \cdot (z \boxplus w) \oplus \Gamma_i \cdot (c \boxplus d)$$

By applying Lemma 4 twice, we see that above approximation has the bias of $\frac{1}{9}(1 + 2^{-2i-2})^2 \approx 2^{-3.2}$ for $i > 0$.

For $k = 1, 2, \ldots, 2^i$, the bias of (14) has the following properties.

- the bias decreases monotonously for $k = 1, 2, \ldots, 2^{i-1}$.
- the bias increases monotonously for $k = 2^{i-1} + 1, \ldots, 2^i$.
- the bias is the highest at $k = 2^i$ and is the lowest (around zero) at $k = 2^{i-1}$.

This bias pattern is repeated for $k = 2^i + 1, \ldots, 2^{i+2} - 1$. If $i > 0$, the overall bias of (14) is around a half of the highest bias, which is $2^{-3.2} * 2^{-1} = 2^{-4.2}$. Hence, the lemma holds.

H Recursion Relation

Let us remind a calculus on recursion relation. Assume that we have the recursive relation $x_n = r \cdot x_{n-1} + c$. If $r \neq 1$, we get $1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$. Thus, x_n can be expressed as $x_n = \frac{c(1-r^n)}{1-r} + x_0 \cdot r^n$. If $r = 1$, then $x_n = x_0 + c \cdot n$.

New Weaknesses in the Keystream Generation Algorithms of the Stream Ciphers TPy and Py*

Gautham Sekar, Souradyuti Paul, and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC,
Kasteelpark Arenberg 10,
B–3001, Leuven-Heverlee, Belgium
{gautham.sekar,souradyuti.paul,bart.preneel}@esat.kuleuven.be

Abstract. The stream ciphers Py, Py6 designed by Biham and Seberry were promising candidates in the ECRYPT-eSTREAM project because of their impressive speed. Since their publication in April 2005, a number of cryptanalytic weaknesses of the ciphers have been discovered. As a result, a strengthened version Pypy was developed to repair these weaknesses; it was included in the category of 'Focus ciphers' of the Phase II of the eSTREAM competition. However, even the new cipher Pypy was not free from flaws, resulting in a second redesign. This led to the generation of three new ciphers TPypy, TPy and TPy6. The designers claimed that TPy would be secure with a key size up to 256 bytes, i.e., 2048 bits. In February 2007, Sekar *et al.* published an attack on TPy with 2^{281} data and comparable time. This paper shows how to build a distinguisher with 2^{275} key/IVs and one outputword per each key (i.e., the distinguisher can be constructed within the design specifications); it uses a different set of weak states of the TPy. Our results show that distinguishing attacks with complexity lower than the brute force exist if the key size of TPy is longer than 275 bits. Furthermore, we discover a large number of similar bias-producing states of TPy and provide a general framework to compute them. The attacks on TPy are also shown to be effective on Py.

1 Introduction

Timeline: the Py-family of Ciphers

– **April 2005.** The ciphers Py and Py6, designed by Biham and Seberry, were submitted to the ECRYPT project for analysis and evaluation in the category of software based stream ciphers [2]. The impressive speed of the cipher Py in software (about 2.5 times faster than the RC4) made it one

* The first author is supported by an IWT SoBeNeT project. The second author is funded by the IBBT (Interdisciplinary Institute for BroadBand Technology), a research institute founded by the Flemish Government in 2004. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 249–262, 2007.

of the fastest and most attractive contestants. The cipher is designed to be used with a key of size 32 bytes (the key size may vary between 1 byte and 256 bytes) and an IV of size 16 bytes (the IV size can vary between 1 and 64 bytes).

- **March 2006 (at FSE 2006).** Paul, Preneel and Sekar reported distinguishing attacks with $2^{89.2}$ data and comparable time against the cipher Py [7]. Crowley [4] later reduced the complexity to 2^{72} by employing a Hidden Markov Model.
- **March 2006 (at the Rump session of FSE 2006).** A new cipher, namely Pypy, was proposed by the designers to rule out the aforementioned distinguishing attacks on Py [3].
- **May 2006 (presented at Asiacrypt 2006).** Distinguishing attacks were reported against Py6 with 2^{68} data and comparable time by Paul and Preneel [8].
- **October 2006 (to be presented at Eurocrypt 2007).** Wu and Preneel showed key recovery attacks against the ciphers Py, Pypy, Py6 with chosen IVs [11]. This attack was subsequently improved by Isobe *et al.* [5].
- **January 2007.** Three new ciphers TPypy, TPy, TPy6 were proposed by the designers [1]. These three ciphers can very well be viewed as the strengthened versions of the previous ciphers Py, Pypy and Py6 where the above attacks do not apply. The ciphers are designed to be secure for any key size between 1 and 256 bytes.
- **February 2007.** Sekar *et al.* published an attack on TPy which requires 2^{281} data and comparable time [9].

In this paper, we show distinguishing attacks on the ciphers TPy and Py with data complexity 2^{275} each. These results outperform the most recent attack on TPy which requires 2^{281} data [9]. However, it is worth noting that the attacks described in [7] can also be applied to TPy. In the design specifications, the TPy and the Py are claimed to be compatible with key size ranging from 8 bits to 2048 bits. If the ciphers are used with key size longer than 275 bits then our attacks are better than exhaustive search. It is also worth noting that the distinguisher can be built within the design specifications of the ciphers. To derive the distinguisher, 2^{275} randomly chosen key/IVs are used and for each of them one outputword is collected. Note that, according to the design specification, TPy can run for 2^{61} rounds (note that each round generates 8 bytes as output) per key where our distinguisher requires only 8 rounds per key.

In addition to the above distinguisher, we detect biases in a large number of outputs at rounds r, $r+2$, t and u where $r > 0$; $t, u \geq 5$; $t \notin \{r, r+2, u\}$; $u \notin \{r, r+2, t\}$. We provide a general framework to compute the biases due to the presence of arbitrarily many weak states. However, we were unable to combine those biases into a more efficient attack. Combining multiple distinguishers into a single and more efficient one is still an alluring open problem.

2 The Round Function of TPy

The round functions of the TPy and the Py are identical. Here, we analyze only the round function of TPy and hence do not describe the key setup and IV setup. Algorithm 1 describes a single round of the TPy. Array P (which is a permutation of $[0, 1, ..., 255]$), array Y (which contains 260 32-bit elements) and the 32-bit variable s are the inputs to the algorithm. Here, 'rotate(A)' denotes a cyclic rotation of the elements of array A by one position. The '$ROTL32(s, k)$' operation means that the 32-bit variable s is rotated to the left by k positions. The output generated in line 5 of the algorithm is labeled 'first output-word' and the output-word of line 6 is labeled 'second output-word'.

Algorithm 1. A Step of TPy

Require: $Y[-3, ..., 256]$, $P[0, ..., 255]$, a 32-bit variable s
Ensure: 64-bit random output
/*Update and rotate P*/
1: swap ($P[0]$, $P[Y[185]\&255]$);
2: rotate (P);
/* Update s*/
3: $s+ = Y[P[72]] - Y[P[239]]$;
4: $s = ROTL32(s, ((P[116] + 18)\&31))$;
/* Output 8 bytes (least significant byte first)*/
5: output $((ROTL32(s, 25) \oplus Y[256]) + Y[P[26]])$;
6: output $((\quad s \quad \oplus Y[-1]) + Y[P[208]])$;
/* Update and rotate Y*/
7: $Y[-3] = (ROTL32(s, 14) \oplus Y[-3]) + Y[P[153]]$;
8: rotate(Y);

3 Notation and Convention

- $O_{a(b)}$ denotes the bth bit ($b = 0$ denotes the least significant bit or lsb) of the first output-word generated at round a. We do not use the second output-word anywhere in our analysis.
- P_a, Y_{a+1} and s_a are the inputs to the algorithm at round a. It is easy to see that when this convention is followed, $O_a = (ROTL32(s_a, 25) \oplus Y_a[256]) + Y_a[P_a[26]]$- the index '$a$' is maintained throughout the expression.
- $Y_a[b]$, $P_a[b]$ denote the bth elements of array Y_a and P_a respectively.
- $Y_a[b]_i$, $P_a[b]_i$ denote the ith bit ($i = 0$ denotes the lsb) of $Y_a[b]$, $P_a[b]$ respectively.
- The operators '+' and '−' denote *addition modulo* 2^{32} and *subtraction modulo* 2^{32} respectively, except when used with expressions which relate two elements of array P. In this case they denote *addition and subtraction over* $\mathbb{Z}$.
- The symbol '$\oplus$' denotes bitwise *exclusive-or* and $\bigcap$ denotes set intersection.
- In $O_{a(i)}$, $s_{a(i)}$ and $Y_a[P_b[X]]_i$, the index representing bit position, i.e., i denotes i mod 32.

– $Y_a^c[P_b[X]]_i$ denotes the complement of $Y_a[P_b[X]]_i$.
– The pseudorandom bit generation algorithm of a stream cipher is denoted by PRBG.

4 Motivational Observations

Our major observation is the detection of a relation between the elements of the internal state and the outputs of the TPy which can, eventually, be used to build a distinguishing attack on the cipher. The relation is outlined in the following theorem.

Theorem 1. $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0$ *if the following* 17 *conditions are simultaneously satisfied.*

1. $P_1[116] \equiv -18 \bmod 32$ *(event* E_1*),*
2. $P_2[116] \equiv 7 \bmod 32$ *(event* E_2*),*
3. $P_3[116] \equiv -4 \bmod 32$ *(event* E_3*),*
4. $P_7[116] \equiv 3 \bmod 32$ *(event* E_4*),*
5. $P_8[116] \equiv 3 \bmod 32$ *(event* E_5*),*
6. $P_1[72] = P_2[239] + 1$ *(event* E_6*),*
7. $P_1[239] = P_2[72] + 1$ *(event* E_7*),*
8. $P_7[72] = P_8[72] + 1$ *(event* E_8*),*
9. $P_7[239] = P_8[239] + 1$ *(event* E_9*),*
10. $P_3[72] = 254$ *(event* E_{10}*),*
11. $P_1[26] = P_3[239] + 2$ *(event* E_{11}*),*
12. $P_1[72] = 3$ *(event* E_{12}*),*
13. $P_3[26] = 0$ *(event* E_{13}*),*
14. $P_1[239] = P_7[26] + 6$ *(event* E_{14}*),*
15. $P_7[153] = 252$ *(event* E_{15}*),*
16. $P_6[153] = P_8[26] + 2$ *(event* E_{16}*),*
17. $d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \oplus e_{8(i+7)} = 0$ *(event* E_{17}*).*[1]

Proof. First, we state and prove two lemmata which will be used to establish the theorem.

Lemma 1. *If*

1. $P_1[116] \equiv -18 \bmod 32$,
2. $P_3[116] \equiv -4 \bmod 32$,
3. $P_7[116] \equiv 3 \bmod 32$,
4. $P_8[116] \equiv 3 \bmod 32$

then the following equations are satisfied:

1. $O_{1(i)} = s_{0(i+7)} \oplus Y_1[P_1[72]]_{i+7} \oplus Y_1^c[P_1[239]]_{i+7} \oplus Y_1[256]_i \oplus Y_1[P_1[26]]_i \oplus c_{1(i)} \oplus d_{1(i+7)}$,
2. $O_{3(i+7)} = s_{2(i)} \oplus Y_3[P_3[72]]_i \oplus Y_3^c[P_3[239]]_i \oplus Y_3[256]_{i+7} \oplus Y_3[P_3[26]]_{i+7} \oplus c_{3(i+7)} \oplus d_{3(i)}$,

[1] The terms c, d, e are the carries generated in certain expressions, the descriptions of which can be found in the proof of Theorem 1.

3. $O_{7(i+7)} = Y_7[P_7[72]]_{i-7} \oplus Y_7^c[P_7[239]]_{i-7} \oplus Y_6[-3]_{i+7} \oplus Y_7[P_7[26]]_{i+7} \oplus Y_6[P_6[153]]_{i+7} \oplus c_{7(i+7)} \oplus d_{7(i-7)} \oplus e_{7(i+7)}$,
4. $O_{8(i+7)} = Y_8[P_8[72]]_{i-7} \oplus Y_8^c[P_8[239]]_{i-7} \oplus Y_7[-3]_{i+7} \oplus Y_8[P_8[26]]_{i+7} \oplus Y_7[P_7[153]]_{i+7} \oplus c_{8(i+7)} \oplus d_{8(i-7)} \oplus e_{8(i+7)}$.

Proof. From Figure 1, we get

$$Y_n[i] = Y_{n+1}[i-1] \tag{1}$$

when $-2 \leq i \leq 256$. When $i = -3$,

$$Y_{n+1}[256] = (ROTL32(s_i, 14) \oplus Y_n[-3]) + Y_n[P_n[153]].$$

Generalizing (1), we have

$$Y_n[i] = Y_{n+k}[i-k] \tag{2}$$

when $-3 \leq i - k \leq 255$. Line 5 of Algorithm 1 gives

$$O_7 = (ROTL32(s_7, 25) \oplus Y_7[256]) + Y_7[P_7[26]]. \tag{3}$$

Let the c_7 denote the carry in the above equation. Since $ROTL32(s_7, 25)_i = s_{7(i-25 \bmod 32)}$,

$$O_{7(i)} = s_{7(i-25 \bmod 32)} \oplus Y_7[256]_i \oplus Y_7[P_7[26]]_i \oplus c_{7(i)}. \tag{4}$$

Lines 3 and 4 of Algorithm 1 give us

$$s_7 = ROTL32(s_6 + Y_7[P_7[72]] - Y_7[P_7[239]], P_7[116] + 18 \bmod 32) \tag{5}$$

$$\Rightarrow s_{7(j)} = s_{6(j-k \bmod 32)} \oplus Y_7[P_7[72]]_{j-k \bmod 32} \oplus Y_7^c[P_7[239]]_{j-k \bmod 32} \oplus d_{7(j-k \bmod 32)} \tag{6}$$

where $k = P_7[116] + 18 \bmod 32$, $d_{7(i)} = f_{7(i)} \oplus g_{7(i)}$ and $d_{7(0)} = 1$ (f_7 and g_7 are the carry terms in (5) which are explained in Sect. 5.2). For simplicity, henceforth we denote $X_{(i \bmod 32)}$ by $X_{(i)}$. Thus (6) becomes,

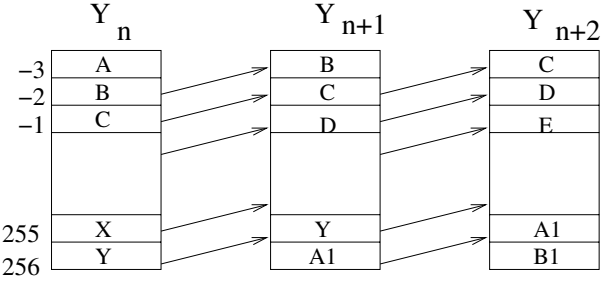


Fig. 1. The figure shows the update of the S-box Y. $Y_n[i] = Y_{n+1}[i-1]$ when $-2 \leq i \leq 256$. $Y_{n+1}[256] = A1$ when $i = -3$ and $A1 = (ROTL32(s_n, 14) \oplus A) + Y_n[P_n[153]]$. Generalizing the above, we can write $Y_n[i] = Y_{n+k}[i-k]$ when $-3 \leq i - k \leq 255$.

$$s_{7(j)} = s_{6(j-k)} \oplus Y_7[P_7[72]]_{j-k} \oplus Y_7^c[P_7[239]]_{j-k} \oplus d_{7(j-k)}. \quad (7)$$

If $j = i - 25 \bmod 32$, then (7) becomes

$$s_{7(i-25)} = s_{6(i-k-25)} \oplus Y_7[P_7[72]]_{i-k-25} \oplus Y_7^c[P_7[239]]_{i-k-25} \oplus d_{7(i-k-25)}. \quad (8)$$

Substituting (8) in (4), we get,

$$O_{7(i)} = s_{6(i-k-25)} \oplus Y_7[P_7[72]]_{i-k-25} \oplus Y_7^c[P_7[239]]_{i-k-25} \oplus Y_7[256]_i \\ \oplus Y_7[P_7[26]]_i \oplus c_{7(i)} \oplus d_{7(i-k-25)}. \quad (9)$$

Next, we have

$$Y_7[256] = (ROTL32(s_6, 14) \oplus Y_6[-3]) + Y_6[P_6[153]], \quad (10)$$

$$Y_7[256]_i = s_{6(i-14)} \oplus Y_6[-3]_i \oplus Y_6[P_6[153]]_i \oplus e_{7(i)} \quad (11)$$

where e_7 is the carry term in (10). Substituting (11) in (9), we get,

$$O_{7(i)} = s_{6(i-k-25)} \oplus s_{6(i-14)} \oplus Y_7[P_7[72]]_{i-k-25} \oplus Y_7^c[P_7[239]]_{i-k-25} \oplus Y_6[-3]_i \\ \oplus Y_7[P_7[26]]_i \oplus Y_6[P_6[153]]_i \oplus c_{7(i)} \oplus d_{7(i-k-25)} \oplus e_{7(i)}. \quad (12)$$

Now, if $k = -11$ (i.e., $k \equiv -11 \bmod 32 \Rightarrow P_7[116] + 18 \equiv -11 \bmod 32 \Rightarrow P_7[116] \equiv 3 \bmod 32$) then $s_{6(i-k-25)} \oplus s_{6(i-14)} = 0$. Hence, when $P_7[116] \equiv 3 \bmod 32$, (12) becomes

$$O_{7(i)} = Y_7[P_7[72]]_{i-14} \oplus Y_7^c[P_7[239]]_{i-14} \oplus Y_6[-3]_i \oplus Y_7[P_7[26]]_i \\ \oplus Y_6[P_6[153]]_i \oplus c_{7(i)} \oplus d_{7(i-14)} \oplus e_{7(i)}. \quad (13)$$

By similar arguments, when $P_8[116] \equiv 3 \bmod 32$,

$$O_{8(i)} = Y_8[P_8[72]]_{i-14} \oplus Y_8^c[P_8[239]]_{i-14} \oplus Y_7[-3]_i \oplus Y_8[P_8[26]]_i \\ \oplus Y_7[P_7[153]]_i \oplus c_{8(i)} \oplus d_{8(i-14)} \oplus e_{8(i)}. \quad (14)$$

From (9), we get

$$O_{1(i)} = s_{0(i-k-25)} \oplus Y_1[P_1[72]]_{i-k-25} \oplus Y_1^c[P_1[239]]_{i-k-25} \oplus Y_1[256]_i \\ \oplus Y_1[P_1[26]]_i \oplus c_{1(i)} \oplus d_{1(i-k-25)}. \quad (15)$$

When $k = 0$ (i.e., $P_1[116] \equiv -18 \bmod 32$), the above equation reduces to

$$O_{1(i)} = s_{0(i+7)} \oplus Y_1[P_1[72]]_{i+7} \oplus Y_1^c[P_1[239]]_{i+7} \oplus Y_1[256]_i \oplus Y_1[P_1[26]]_i \\ \oplus c_{1(i)} \oplus d_{1(i+7)}. \quad (16)$$

Similarly, when $P_3[116] \equiv -4 \bmod 32$, we have

$$O_{3(i+7)} = s_{2(i)} \oplus Y_3[P_3[72]]_i \oplus Y_3^c[P_3[239]]_i \oplus Y_3[256]_{i+7} \oplus Y_3[P_3[26]]_{i+7} \\ \oplus c_{3(i+7)} \oplus d_{3(i)}. \quad (17)$$

From (13) and (14), we derive the following results:

$$O_{7(i+7)} = Y_7[P_7[72]]_{i-7} \oplus Y_7^c[P_7[239]]_{i-7} \oplus Y_6[-3]_{i+7} \oplus Y_7[P_7[26]]_{i+7} \oplus Y_6[P_6[153]]_{i+7} \oplus c_{7(i+7)} \oplus d_{7(i-7)} \oplus e_{7(i+7)}, \quad (18)$$

$$O_{8(i+7)} = Y_8[P_8[72]]_{i-7} \oplus Y_8^c[P_8[239]]_{i-7} \oplus Y_7[-3]_{i+7} \oplus Y_8[P_8[26]]_{i+7} \oplus Y_7[P_7[153]]_{i+7} \oplus c_{8(i+7)} \oplus d_{8(i-7)} \oplus e_{8(i+7)}. \quad (19)$$

This completes the proof. □

Now we state the second lemma.

Lemma 2. *$s_{0(i+7)} = s_{2(i)}$ if the following conditions are simultaneously satisfied,*

1. $P_1[116] \equiv -18 \bmod 32$,
2. $P_2[116] \equiv 7 \bmod 32$,
3. $P_1[72] = P_2[239] + 1$,
4. $P_1[239] = P_2[72] + 1$.

Proof. Equation (5) gives us:

$$s_1 = ROTL32(s_0 + Y_1[P_1[72]] - Y_1[P_1[239]], P_1[116] + 18 \bmod 32).$$

The first condition ($P_1[116] \equiv -18 \bmod 32$) reduces this to

$$s_1 = s_0 + Y_1[P_1[72]] - Y_1[P_1[239]].$$

Therefore,

$$s_2 = ROTL32(s_0 + Y_2[P_2[72]] - Y_2[P_2[239]] + Y_1[P_1[72]] - Y_1[P_1[239]], P_2[116] + 18 \bmod 32).$$

Conditions 3 and 4 reduce the above equation to

$$s_2 = ROTL32(s_0, P_2[116] + 18 \bmod 32).$$

Finally, with condition 2 (i.e., $P_2[116] \equiv 7 \bmod 32$), the previous equation becomes

$$\begin{aligned} s_2 &= ROTL32(s_0, 25) \\ \Rightarrow s_{2(i)} &= ROTL32(s_0, 25)_i = s_{0(i-25)} \\ &= s_{0(i+7)}. \end{aligned} \quad (20)$$

This completes the proof. □

Now we observe that, when the conditions listed under (i) Lemma 1 (i.e., events E_1, E_3, E_4 and E_5) and (ii) Lemma 2 (i.e., events E_1, E_2, E_6 and E_7) are simultaneously satisfied, then the expression $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$ is the XOR of the terms which are listed in Table 1 (grouped according to the bit positions).[2] Similarly, the 'carries' in Table 1 are elaborated in Table 2.

[2] Note that none of the terms listed in Table 1 is of the form A^c because we used the fact that $A^c \oplus B^c = A \oplus B$ in (16), (17), (18) and (19).

Table 1. Terms generated in $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$, when events E_1 to E_7 simultaneously occur, grouped by their bit positions

Bit position: $i-7$	**Bit position:** i	**Bit position:** $i+7$
$Y_7[P_7[72]]$	$Y_1[256]$	$Y_1[P_1[72]]$
$Y_7[P_7[239]]$	$Y_1[P_1[26]]$	$Y_1[P_1[239]]$
$Y_8[P_8[72]]$	$Y_3[P_3[72]]$	$Y_3[256]$
$Y_8[P_8[239]]$	$Y_3[P_3[239]]$	$Y_3[P_3[26]]$
Carries	Carries	$Y_6[P_6[153]]$
		$Y_6[-3]$
		$Y_7[P_7[26]]$
		$Y_7[P_7[153]]$
		$Y_7[-3]$
		$Y_8[P_8[26]]$
		Carries

Table 2. Carry terms generated in $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$ grouped by their bit positions

Bit position: $i-7$	**Bit position:** i	**Bit position:** $i+7$
d_7	c_1	d_1
d_8	d_3	c_3
		c_7
		e_7
		c_8
		e_8

If the Y-terms in Table 1 are pairwise equated (this is achieved when the events E_8 through to E_{16} occur) then we get

$$\begin{aligned} O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = {} & d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \\ & \oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \\ & \oplus e_{8(i+7)}. \end{aligned} \tag{21}$$

Now, when the RHS of (21) equals zero (i.e., E_{17} occurs) we get

$$O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0.$$

This completes the proof. □

5 Computation of the Bias

In this section, we quantify the bias in the outputs of TPy induced by the fortuitous events similar to the one described in Sect. 4. Now it is important to note that there may be *more than one set of 17 conditions* possible, where each

of them results in $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0$ (let us assume that there are n such sets). In Theorem 1, we listed one such set. Our experiments suggest that these n sets are mutually independent, however, a formal proof of that is nontrivial.

Each of the events E_1 to E_5 occurs with approximate probability $\frac{1}{32}$ and each of the events E_6 to E_{16} occurs with probability which is approximately $\frac{1}{256}$. Let p denote the probability that condition 17 is satisfied. Let F denote the event $\bigcap_{j=1}^{16} E_j$. Therefore,

$$P[F] = (\frac{1}{32})^5 \cdot (\frac{1}{256})^{11}.$$

We see that there are n F-like events (i.e., the intersection of 16 conditions). Let F_n denote the union of these n events. Since, each event occurs with approximately the same probability,

$$\begin{aligned} P[F_n] &\approx n \cdot P[F] \\ &\approx n \cdot (\frac{1}{32})^5 \cdot (\frac{1}{256})^{11} \\ &= n \cdot \frac{1}{2^{113}}. \end{aligned}$$

From Table 1, we get the maximum number of ways that terms of a particular column can be pairwise equated and hence the upper bound on n can be calculated to be $\binom{10}{2} \cdot 3 \cdot 3 = 405$.

5.1 Formulating the Bias

Now, we establish a formula to compute $P[O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0]$, under the assumption of a perfectly random key/IV setup and the uniformity of bits when F_n does not occur.Our experiments suggest that it is infeasible to find a set of conditions such that the overall bias (computed on the basis of the aforementioned assumption of randomness in the event that F_n does not occur) is canceled or reduced in magnitude. Therefore, this assumption is reasonable. Let T denote $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$. Then using Bayes' rule we get

$$\begin{aligned} P[T=0] &= P[T=0|F_n \cap E_{17}] \cdot P[F_n \cap E_{17}] + P[T=0|F_n^c \cup E_{17}^c] \cdot P[F_n^c \cup E_{17}^c] \\ &= P[T=0|F_n \cap E_{17}] \cdot P[F_n \cap E_{17}] + P[T=0|F_n^c \cap E_{17}] \cdot P[F_n^c \cap E_{17}] \\ &+ P[T=0|F_n \cap E_{17}^c] \cdot P[F_n \cap E_{17}^c] + P[T=0|F_n^c \cap E_{17}^c] \cdot P[F_n^c \cap E_{17}^c] \\ &= 1 \cdot (n \cdot p \cdot \frac{1}{2^{113}}) + \frac{1}{2} \cdot (1 - n \cdot \frac{1}{2^{113}}) \cdot p + 0 \cdot P[F_n \cap E_{17}^c] \\ &+ \frac{1}{2} \cdot (1 - n \cdot \frac{1}{2^{113}}) \cdot (1-p) \\ &= \frac{1}{2} + n \cdot (2p-1) \cdot \frac{1}{2^{114}}. \end{aligned} \tag{22}$$

Hence, we see that the distribution of the outputs $(O_{1(i)}, O_{3(i+7)}, O_{7(i+7)}, O_{8(i+7)})$ is biased. The bias is equal to $n \cdot (2p-1) \cdot \frac{1}{2^{114}}$. In the following section,

Table 3. Truth table for computing p_i (NR=Not Required)

$c_{(i-1)}$	$S_{(i-1)}$	$X_{(i-1)}$	$Z_{(i-1)}$	$c_{(i)}$	Probability
0	0	0	0	0	$\frac{p_{i-1}}{8}$
0	0	0	1	0	$\frac{p_{i-1}}{8}$
0	0	1	0	0	$\frac{p_{i-1}}{8}$
0	0	1	1	0	$\frac{p_{i-1}}{8}$
0	1	0	0	0	$\frac{p_{i-1}}{8}$
0	1	0	1	1	NR
0	1	1	0	1	NR
0	1	1	1	0	$\frac{p_{i-1}}{8}$
1	0	0	0	0	$\frac{1-p_{i-1}}{8}$
1	0	0	1	1	NR
1	0	1	0	1	NR
1	0	1	1	0	$\frac{1-p_{i-1}}{8}$
1	1	0	0	1	NR
1	1	0	1	1	NR
1	1	1	0	1	NR
1	1	1	1	1	NR

we provide formulas to compute p, i.e., the probability that E_{17} occurs; or more generally, the probability that the 17th condition of each of the n F-like events occurs, i.e., $P[d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \oplus e_{8(i+7)}] = 0$.

5.2 Biases in the Carry Terms

In this section, we provide formulas to calculate the bias in the carry terms. The carry terms c and e are generated in expressions of the form $(S \oplus X) + Z$. We now proceed to calculate $P[c_{l(i)} = 0]$ assuming that S, X and Z are uniformly distributed and independent. Under this assumption, $P[S_i = 0] = P[X_i = 0] = P[Z_i = 0] = \frac{1}{2}$, that is, the probability that the carry bit at position i equals zero depends only on i. Stated otherwise, $P[c_{(i)} = 0] = P[e_{(i)} = 0]$. Let $P[c_{(i)} = 0]$ be denoted by p_i. Since there is no carry on the lsb, $p_0 = 0$. We now have Table 3.

From Table 3, using Bayes' rule we get

$$p_i = \frac{p_{i-1}}{2} + \frac{1}{4}.$$

Solving this recursion, given $p_0 = 0$, we get

$$p_i = \frac{1}{2} + \frac{1}{2^{i+1}}. \tag{23}$$

Now, the carry terms f and g are generated in expressions of the form $S+X-Z$. This can be rewritten as $S+X+Z^c+1$ since the additions in these two expressions are modulo 2^{32}. The presence of two carries in $S+X+Z$ is demonstrated using the Figure 2. The carries generated in $S + X + Z^c + 1$ can be thought of as

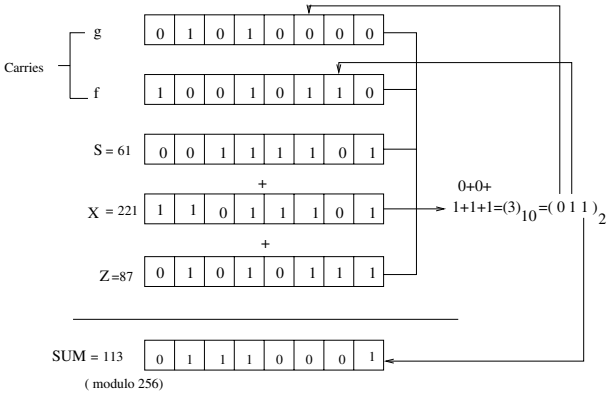


Fig. 2. An example showing how the carries are generated when three 8-bit variables $S = 61$, $X = 221$ and $Z = 87$ are added

carries generated in $S + X + A$ where $A = Z^c$ and the carries on the lsb $f_{(0)} = 1$, $g_{(0)} = 0$. Let q_i denote $P[f_{(i)} = 0]$ and r_i denote $P[g_{(i)} = 0]$. Hence, $q_0 = 0$, $r_0 = 1$ and $r_1 = 1$. Now we have Table 4.

From Table 4, using Bayes' rule we get

$$q_i = \frac{1}{2} + \frac{5 \cdot q_{i-1} \cdot r_{i-1}}{8} - \frac{q_{i-1}}{4} - \frac{r_{i-1}}{4}, \tag{24}$$

$$r_{i+1} = \frac{1}{2} - \frac{q_{i-1} \cdot r_{i-1}}{4} + \frac{3 \cdot q_{i-1}}{8} + \frac{3 \cdot r_{i-1}}{8}. \tag{25}$$

Using the initial conditions, $q_0 = 0$, $r_0 = 1$ and $r_1 = 1$, q_i and r_i are computed recursively. Since $d_{m(i)}$ denotes $f_{m(i)} \oplus g_{m(i)}$ for any $m > 0$,

1. $P[d_{7(i-7)} = 0] = P[d_{8(i-7)} = 0] = q_{i-7 \bmod 32} \cdot r_{i-7 \bmod 32} + (1 - q_{i-7 \bmod 32}) \cdot (1 - r_{i-7 \bmod 32})$,
2. $P[c_{1(i)} = 0] = \frac{1}{2} + \frac{1}{2^{i+1}}$,
3. $P[d_{3(i)} = 0] = q_i \cdot r_i + (1 - q_i) \cdot (1 - r_i)$,
4. $P[d_{1(i+7)} = 0] = q_{i+7 \bmod 32} \cdot r_{i+7 \bmod 32} + (1 - q_{i+7 \bmod 32}) \cdot (1 - r_{i+7 \bmod 32})$,
5. $P[c_{3(i+7)} = 0] = P[c_{7(i+7)} = 0] = P[e_{7(i+7)} = 0] = P[c_{8(i+7)} = 0] = P[e_{8(i+7)} = 0] = \frac{1}{2} + \frac{1}{2^{(i+7 \bmod 32)+1}}$.

Using the above formulas, the value of p can be computed for any given i. Running simulation, we find that the maximum bias in the chosen outputs occurs when $i = 25$ which corresponds to $p = 0.5 - 2^{-34.2}$. Hence, (22) gives us

$$P[T = 0] = \frac{1}{2} - \frac{n}{2^{147.2}}$$

$$\Rightarrow P[T = 1] = \frac{1}{2} + \frac{n}{2^{147.2}},$$

when $i = 25$. Substituting $n = 405$ in the above equation, we get:

$$P[T=1] = \frac{1}{2} + \frac{1}{2^{138.5}}. \tag{26}$$

This is an upper bound on the probability that the outputs ($O_{1(i)}$, $O_{3(i+7)}$, $O_{7(i+7)}$, $O_{8(i+7)}$) of TPy are biased. From Sect. 4, we found that $n \geq 1$. From the previous discussion, we see that $n \leq 405$. Hence, $1 \leq n \leq 405$. If $n = 1$, then $P[T=1] = \frac{1}{2} + \frac{1}{2^{147.2}}$. Thus,

$$\frac{1}{2}(1 + \frac{1}{2^{146.2}}) \leq P[T=1] \leq \frac{1}{2}(1 + \frac{1}{2^{137.5}}). \tag{27}$$

Table 4. Truth table for computing q_i and r_{i+1} using q_{i-1} and r_{i-1} (NR=Not Required)

$f_{(i-1)}$	$g_{(i-1)}$	$S_{(i-1)}$	$X_{(i-1)}$	$Z_{(i-1)}$	$f_{(i)}$	$g_{(i+1)}$	Probability
0	0	0	0	0	0	0	$\frac{q_{i-1} \cdot r_{i-1}}{8}$
0	0	0	0	1	0	0	$\frac{q_{i-1} \cdot r_{i-1}}{8}$
0	0	0	1	0	0	0	$\frac{q_{i-1} \cdot r_{i-1}}{8}$
0	0	0	1	1	1	0	NR
0	0	1	0	0	0	0	$\frac{q_{i-1} \cdot r_{i-1}}{8}$
0	0	1	0	1	1	0	NR
0	0	1	1	0	1	0	NR
0	0	1	1	1	0	0	$\frac{q_{i-1} \cdot r_{i-1}}{8}$
0	1	0	0	0	0	0	$\frac{q_{i-1} \cdot (1-r_{i-1})}{8}$
0	1	0	0	1	1	0	NR
0	1	0	1	0	1	0	NR
0	1	0	1	1	1	0	NR
0	1	1	0	0	1	0	NR
0	1	1	0	1	1	0	NR
0	1	1	1	0	1	0	NR
0	1	1	1	1	0	1	$\frac{q_{i-1} \cdot (1-r_{i-1})}{8}$
1	0	0	0	0	0	0	$\frac{(1-q_{i-1}) \cdot r_{i-1}}{8}$
1	0	0	0	1	1	0	NR
1	0	0	1	0	1	0	NR
1	0	0	1	1	1	0	NR
1	0	1	0	0	1	0	NR
1	0	1	0	1	1	0	NR
1	0	1	1	0	1	0	NR
1	0	1	1	1	0	1	$\frac{(1-q_{i-1}) \cdot r_{i-1}}{8}$
1	1	0	0	0	1	0	NR
1	1	0	0	1	1	0	NR
1	1	0	1	0	1	0	NR
1	1	0	1	1	0	1	$\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$
1	1	1	0	0	1	0	NR
1	1	1	0	1	0	1	$\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$
1	1	1	1	0	0	1	$\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$
1	1	1	1	1	0	1	$\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$

6 The Distinguisher

A distinguisher is an algorithm which distinguishes a given stream of bits from a stream of bits generated by a perfect PRBG. The distinguisher is constructed by collecting sufficiently many outputs $(O_{1(25)}, O_{3(0)}, O_{7(0)}, O_{8(0)})$ generated by as many key/IVs. To compute the minimum number of samples required to establish the distinguisher, we use the following corollary of a theorem from [6].

Corollary 1. *If an event e occurs in a distribution X with probability p and in Y with probability $p(1+q)$ then, if $p = \frac{1}{2}$, $O(\frac{1}{q^2})$ samples are required to distinguish X from Y with non-negligible probability of success.*

In the present case, e is the event $O_{1(25)} \oplus O_{3(0)} \oplus O_{7(0)} \oplus O_{8(0)} = 0$, X is the distribution of the outputs O_1, O_3, O_7 and O_8 produced by a perfectly random keystream generator and Y is the distribution of the outputs produced by TPy. From (27), $p = \frac{1}{2}$ and the highest value of $q = \frac{1}{2^{137.5}}$. Hence $O(\frac{1}{(2^{-137.5})^2}) = O(2^{275})$ output samples are needed to construct the best distinguisher with a non-negligible probability of success. Note that this is an improvement by a factor of 2^6 over the data complexity of 2^{281} obtained in [9].

7 A Family of Distinguishers

In Sect. 4 we found that the outputs at rounds 1, 3, 7 and 8 are biased allowing us to build a distinguisher. It is found that there exist plenty of 4-tuples of biased outputs. The generalization is presented in the following theorem.

Theorem 2. *The distribution of the outputs $(O_{r(i)}, O_{r+2(i+7)}, O_{t(i+7)}, O_{u(i+7)})$ of the TPy are biased for many suitably chosen (r, t, u)'s where $r > 0$; $t, u \geq 5$; $t \notin \{r, r+2, u\}$; $u \notin \{r, r+2, t\}$.*

The proof is similar to the proof furnished for Theorem 1, however, a detailed proof has been provided in the Appendix A of [10]. This allows us to construct a family of distinguishers for the cipher TPy. It seems possible to combine these huge number of distinguishers in order to construct one single efficient distinguisher; however, any concrete mathematical model to combine them is still an interesting open problem. Another major implication of the above generalization theorem is the fact that the TPy outputs will remain always biased no matter how many initial outputwords are discarded from the keystream.

8 Attacks on Py

The PRBG of the cipher Py is identical with that of TPy. The attacks described in the previous sections exploit the weaknesses in the PRBG of TPy only. Therefore, all the attacks are applicable to Py also.

9 Conclusion and Open Problems

The paper develops a family of distinguishers from the outputs $(O_{r(i)}, O_{r+2(i+7)}, O_{t(i+7)}, O_{u(i+7)})$ of TPy (and Py), where $r > 0$; $t, u \geq 5$; $t \notin \{r, r+2, u\}$; $u \notin \{r, r+2, t\}$. Note that the TPy is one of the strongest members of the Py-family of ciphers. The best distinguisher works with data complexity 2^{275} which records an improvement of a factor of 64 over the previous attack. In addition, we detect a large number of bias-producing states of TPy and compute them in a general framework. It is reasonable to assume that these weak states can be combined to mount a more efficient attack on TPy; however, methods to combine many distinguishers into a single yet more efficient one is still an open problem.

Acknowledgments

We thank the anonymous reviewers of ISC'07 for their constructive comments on our work.

References

1. Biham, E., Seberry, J.: Tweaking the IV Setup of the Py Family of Ciphers – The Ciphers Tpy, TPypy, and TPy6 (January 25, 2007), Published on the author's webpage at http://www.cs.technion.ac.il/~biham/
2. Biham, E., Seberry, J.: Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays (ecrypt submission 2005)
3. Biham, E., Seberry, J.: Pypy (Roopy): Another Version of Py. (ecrypt submission 2006)
4. Crowley, P.: Improved Cryptanalysis of Py. In: Workshop Record of SASC, - Stream Ciphers Revisited, ECRYPT Network of Excellence in Cryptology, February 2006, Leuven, Belgium, pp. 52–60 (2006)
5. Isobe, T., Ohigashi, T., Kuwakado, H., Morii, M.: How to Break Py and Pypy by a Chosen-IV Attack. eSTREAM, ECRYPT Stream Cipher Project, Report2006/060
6. Mantin, I., Shamir, A.: A Practical Attack on Broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2002)
7. Paul, S., Preneel, B., Sekar, G.: Distinguishing Attacks on the Stream Cipher Py. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 405–421. Springer, Heidelberg (2006)
8. Paul, S., Preneel, B.: On the (In)security of Stream Ciphers Based on Arrays and Modular Addition. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 69–83. Springer, Heidelberg (2006)
9. Sekar, G., Paul, S., Preneel, B.: Weaknesses in the Pseudorandom Bit Generation Algorithms of the Stream Ciphers TPypy and TPy, available at http://eprint.iacr.org/2007/075.pdf
10. Sekar, G., Paul, S., Preneel, B.: New Weaknesses in the Keystream Generation Algorithms of the Stream Ciphers TPy and Py, available at http://eprint.iacr.org/2007/230.pdf
11. Wu, H., Preneel, B.: Differential Cryptanalysis of the Stream Ciphers Py, Py6 and Pypy. In: Naor, M. (ed.) Eurocrypt 2007. LNCS, vol. 4515, pp. 276–290. Springer, Heidelberg (2007)

Queue Management as a DoS Counter-Measure?

Daniel Boteanu[1], José M. Fernandez[1], John McHugh[2], and John Mullins[1]

[1] École Polytechnique de Montréal
[2] Dalhousie University

Abstract. In this paper, we study the performance of timeout-based queue management practices in the context of flood denial-of-service (DoS) attacks on connection-oriented protocols, where server resources are depleted by uncompleted illegitimate requests generated by the attacker. This includes both crippling DoS attacks where services become unavailable and Quality of Service (QoS) degradation attacks. While these queue management strategies were not initially designed for DoS attack protection purposes, they do have the desirable side-effect or providing some protection against them, since illegitimate requests time out more often than legitimate ones. While this fact is intuitive and well-known, very few quantitative results have been published on the potential impact on DoS-attack resilience of various queue management strategies and the associated configuration parameters. We report on the relative performance of various queue strategies under a varying range of attack rates and parameter configurations. We hope that such results will provide usable configuration guidelines for end-server or network appliance queue hardening. The use of such optimisation techniques is complementary to the upstream deployment of other types of DoS-protection countermeasures, and will probably prove most useful in scenarios where some residual attack traffic still bypasses them.

Keywords: Denial of service attack, degradation of service attack, queue management, timeout, dynamic timeout.

1 Introduction

A denial-of-service (DoS) network attack occurs when the victim receives a malicious stream of packets that prevent the legitimate communication from taking place. DoS flood attacks consist in sending the victim (typically a server) a higher volume of traffic than it can handle. This can be achieved either by saturating the server's network connection or by using weaknesses in the communication protocols that typically allow the attacker to generate high server resource usage for a limited attacker effort. Distributed denial-of-service (DDoS) attacks are simply DoS attacks performed by multiple agents, most frequently simultaneously. In this paper we direct our attention towards the resource exhaustion attacks on connection oriented protocols. Although the studied-to-death SYN-flood attack fits well into this category, we use it merely as an example to explain our approach. As we will discuss, it is our hope that our approach could

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 263–280, 2007.

potentially be applied to other TCP-based attacks (e.g. ACK flood) or higher-level attacks against Web servers (straight HTTP or via SSL), FTP servers, VPN gateways, mail servers, or even DoS protection mechanisms in upstream network appliances.

The impact of a DoS attack on a particular system will vary depending on the protocols and applications involved. Furthermore, an attack can have measurable impact on the Quality of Service (QoS) of a system even when the server resources are not completely exhausted [21], such as in the case of *Degradation of Service* attacks [19]. While degrading QoS or even rendering a service unavailable might be possible, this always comes at a cost for the attacker. For a given service level or attack impact, there is a direct relationship between the resources expended by the attacker and the target. These tradeoffs have been discussed for crippling DoS attacks [16,17], but these formalisms cannot be easily applied to QoS degradation attacks. While some experimental testbeds have been proposed to try to measure these tradeoffs [2], there are in fact very few quantitative results (modelling or experimental) concerning degradation of service attacks.

Various methods and appliances for protecting against DoS attacks have been suggested, for example Cisco Guard XT, Captus IPS, COSSACK, DefCOM, D-WARD, MANAnet Shield, Mazu Enforcer, NetBouncer, Peakflow, Proof of Work, Pushback, Secure Overlay Services, Traceback and others (see [19,20] for complete surveys on the topic). These can be viewed as first- and second-line defences, where first-line defences use traffic profiling or anomaly detection mechanisms and filter it accordingly [8,25,22], and second-line defences consist in modifying TCP/IP protocols to positively affect the resource tradeoff in favour of the defender [3,28,10,12,7].

Nonetheless, it is possible for sophisticated attacks to evade both types of defences. Thus, a considerable amount of residual attack traffic could still evade both network- and host-based defences and reach the end server OS and application connection queues. When all traffic-discriminating counter-measures have been bypassed, legitimate and residual attack traffic is indistinguishable. Fortunately, certain features of the end server can mitigate the impact of residual traffic, even in those conditions. These features therefore constitute a last line of defence. Queue management algorithms that were initially designed to minimise the impact of network traffic loss or high latency fall into that category. One of the most common such features is the attribution of timeout periods to all incoming connections. Protocols that implement this feature and that do not necessarily require explicit messages to close a connection are called *soft protocols* and they perform better than their counterpart *hard protocols* in unexpected network conditions like DoS attacks [14]. In practice, the timeouts can be adjusted dynamically according to administrator-configurable thresholds on resource usage levels, as has been suggested and implemented in network appliances [10] and OS [18] for the specific purpose of improving resiliency against DoS attacks. When these thresholds are reached, the server is placed in a "protective" state, which in principle has the effect of favouring fast legitimate connections over attack connections.

Unfortunately, it is not clear at all what the "optimal" threshold values are, as there is no quantitative method for estimating the parameters that minimise the effects of DoS attacks while maintaining equivalent levels of QoS. In the rest of this paper, we try to address this gap. One of the reasons we are interested in this is because such features are very general and already in place at the various network and application layers. Note also that maximising their effectiveness as DoS protections is complementary and in principle compatible with the deployment and use of other upstream defences.

In the next section we propose a stochastic modelling tool for DoS attacks, based on Markov chains. Using this model, we analyse three different timeout-based protection strategies in Sect. 3. We provide in Sect. 4 the experimental results obtained by simulating these strategies according to two different implementations. Finally, we conclude and give directions for future work in Sect. 5.

2 Modelling Servers Under DoS Attack with Markov Chains

Markov chains are a stochastic modelling tool that describe the states and dynamics of a system at successive times. They are said to be *memoryless* if the probability of transition between any two states is independent of the previous states. The stochastic process generating state transition events is thus said to be *markovian*, which is equivalent to saying that they are distributed in time according to a Poisson distribution. Markov chains can also be used to model systems in which this is not the case, i.e. those where state transitions probabilities will depend on past history. In instances where the key parameters such as rate of arrivals and departures are known, the model can be "solved". First, this means that given state probabilities at given time, predictions can be made about state probabilities at a later time. We can also compute *steady-state probabilities*, which correspond to the likelihood of the various states at the equilibrium of the system.

Markov chains are suitable for modelling network performance and have been used in that purpose for many years. In particular, Markov Chains have also been used as modelling tool in network security. Baras [1] suggests detecting route falsification attacks in mobile ad-hoc networks (MANET) using a Hidden Markov Model (HMM). More recent studies [27] show that using edge sampling techniques along with HMM can be used to reconstruct a network attack path. HMMs can also be used in Intrusion Detection Systems [11,15], the transitions between each state in the Markov model being generated by intrusion, detection and recovery events. Finally, Khan *et al.* [13] have successfully used Markov Chain modelling of queues to design DoS traffic detection strategies. In our case, we will use Markov chains to model the performance of servers under DoS attacks.

2.1 Description of the Model

Typical Markov chain models used in network performance have each state characterised by the number of connections in the system. A maximum number of

connections c can be served at the same time. Connections arrive with rate λ and are served with rate μ. In our case, each state in the chain is characterised by two values: N_l and N_m, the number of connections used by legitimate users and malicious users, respectively. A maximum number of both legitimate and malicious connection c can be served in the same time. All connection requests that arrive when the server is in a saturated state ($N_l + N_m = c$) will be rejected. Transitions between states occur with different rates for the legitimate and malicious connection requests: λ_l and μ_l for the arrivals and servings of legitimate connections and λ_m and μ_m for the arrivals and servings of malicious connections. The chain has a triangular form where states on the upper line represent that no malicious connections are present in the system and states on the diagonal represent that only malicious connections are present in the system (see Fig. 6 in the appendix). The following events generate transitions between states:

- *connection arrived*: the server received a connection request from a client. It occurs at a rate λ;
- *connection completed*: the connection was either elevated to a higher-level protocol, or the client was served with the required information and the connection was closed successfully. It occurs at a rate μ_l;
- *connection rejected*: the server was not able to serve the connection because no more connections channels were available (queue full). It occurs at a rate ϕ_r;
- *connection expired*: the server tried to serve the connection but the communication timed out and the connection was dropped. It occurs at a rate ϕ_e; and
- *connection failed*: the connection was either rejected or it expired. It occurs at a rate $\phi = \phi_r + \phi_e$.

In the particular case of a SYN-flood attack, N_l will actually represent the number of legitimate connections (and N_m the number of malicious ones) that are half-open. The *connection arrived* and *connection completed* events represent a SYN message and the corresponding ACK message being received by the server, respectively. In the case of an SSL connection depletion attack, N_l and N_m represent the number of legitimate and malicious completed TCP connections, respectively, that have not yet established a secure channel and for which the negotiation phase is still in progress. The *connection arrived* events represent the *Client hello* message being received by the server and the *connection completed* events represent the corresponding *Finished* message being sent by the server. It is even possible to consider a nested model, each level representing a different layer in the protocol stack.

How realistic is this model regarding legitimate arrival and service rates? It is known that user sessions initiations resemble phone calls [23] and thus have a Poisson arrival process with exponential inter-arrival times. We will make the supposition that all incoming connections follow this pattern, assumption that is used in other DoS related research [21,5]. In most cases, serving rates depend only of network transit times but in some cases user interaction is also a factor.

It has been shown that because of the network queueing algorithms, all the IP packet traffic tends toward a Poisson process as the load increases [4]. For modelling purposes, we will make the supposition that the network is heavily used and therefore that legitimate service rate follows a Poisson process model. According to our assumption, *connections completed* events are generated at exponential intervals of time from the *connection arrived* events, if the timeout has not elapsed. Otherwise, *connection expired* events are generated at timeout intervals from the *connection arrived* events.

The rate at which *connection completed* messages are generated by the legitimate clients is μ_c. Only messages that arrive to the server before their timeout elapses will generate *connection completed* events; we thus have that $\mu_l \geq \mu_c$. All the other will be ignored by the server and *connection expired* events are generated when the timeout elapses. Therefore, the Probability Distribution Function (PDF) of the legitimate connection service time $G_l(t)$ will have the form of an exponential distribution for t smaller than the timeout t_{out}, followed by an appropriately weighted delta Dirac function at t_{out}

$$G_l(t) = \begin{cases} \mu_c e^{-t\mu_c} & t < t_{\text{out}} \\ \delta(t - t_{\text{out}}) p_{\text{expire}} & t = t_{\text{out}} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where

$$p_{\text{expire}} = \int_{t_{\text{out}}}^{\infty} \mu_c e^{-t\mu_c} dt = e^{-t_{\text{out}}\mu_c} \,. \tag{2}$$

The mean service time and the service rate for legitimate connections are

$$t_l \triangleq \int_0^{\infty} t G_l(t) dt = \frac{1 - e^{-t_{\text{out}}\mu_c}}{\mu_c} \,; \qquad \mu_l \triangleq \frac{1}{t_l} = \frac{\mu_c}{1 - e^{-t_{\text{out}}\mu_c}} \tag{3}$$

While we model legitimate packet arrivals as a Poisson process this is not general for attack traffic as the attacker is free to use whatever strategy he or she wants. Even though there is no proof that this is optimal, the attacker might want to mimic the legitimate arrivals process in order to thwart certain time analysis detection methods. In any case, we will assume that the *residual attack traffic*, unfiltered by upstream defence mechanisms is distributed according to a Poisson distribution, because otherwise it could have been potentially discriminated by such techniques. We make this assumption in order to be able to construct a simple enough mathematical model that we can numerically resolve. However, we will later explore in Sect. 4 attacks for which this is not true.

Concerning the malicious packet service process, the strategy of the attacker is to exhaust the server resources using the smallest effort possible. This is achieved by generating the *connection arrived* events and then abandoning the communication without any notice to the server. Malicious connections will eventually all expire and generate *connection expired* events at t_{out} intervals of time from

the *connection arrived* events. The malicious connection service rate is in this case $\mu_m = 1/t_{\text{out}}$.

Although the triangular DoS Markov chain model that we presented describes the states in which the server will be during an attack, these states are not directly visible because individual connections can not be labelled as legitimate or malicious. For this reason, we will analyse the *visible* Markov chain that has $c+1$ states, each state being characterised by the number of connections N used by both legitimate and malicious users. The probability that the visible Markov chain is in a state N is the sum of all probabilities that the hidden Markov chain is in state (N_l, N_m) with $N_l + N_m = N$.

The visible connection arrival process is the sum of two Poisson processes with rates λ_l and λ_m and thus also a Poisson process with rate $\lambda = \lambda_l + \lambda_m$. In a Markov chain model the *load* is defined as the ratio between the arrival and service rates. In our case, we distinguish the load generated by the legitimate users $\rho_l = \lambda_l/\mu_l$, and the load generated by malicious users $\rho_m = \lambda_m/\mu_m$. The overall load ρ cannot be computed directly because the service processes are not memoryless. Our goal is to compute the overall load by approximating the overall mean service time $\tilde{t}$. We consider $\tilde{t}$ to be constant in time and equal to the average of the mean legitimate service time t_l and mean malicious service time t_m weighted by the legitimate load and the malicious load, respectively:

$$\tilde{t} = \frac{\rho_l}{\rho_l + \rho_m} t_l + \frac{\rho_m}{\rho_l + \rho_m} t_{\text{out}} \tag{4}$$

The approximative mean service rate in the visible chain is:

$$\tilde{\mu} \triangleq \frac{1}{\tilde{t}} = \frac{\mu_l \mu_m (\lambda_m \mu_l + \lambda_l \mu_m)}{\lambda_m \mu_l^2 + \lambda_l \mu_m^2} \tag{5}$$

We can now calculate an approximative overall load generated by both legitimate and malicious users as $\tilde{\rho} \triangleq \lambda/\tilde{\mu}$. With this approximation we can compute the steady-state probability that the system is in the state k using Erlang's loss formula:

$$p_k = \frac{\tilde{\rho}^k}{k!} / \sum_{i=0}^{c} \frac{\tilde{\rho}^i}{i!} \tag{6}$$

2.2 Approximate Solutions to the Model

Because the connections are served independently, the only significant performance measure is the probability ϕ that a legitimate connection will fail, which is equal to the probability that the connection will be rejected ϕ_r plus the probability the connection will expire ϕ_e, i.e. $\phi = \phi_r + \phi_e$.

The *blocking probability* is by definition the probability that the system is saturated, i.e. that the queue is full. A connection is rejected if the server is

saturated when the *connection arrived* event is generated. The probability that a connection is rejected ϕ_r is thus equal to the probability that the server is in state c at that moment. If the system were at equilibrium, this will be exactly the steady-state probability p_c. If we assume that the system will never be far from equilibrium, we can approximate it as such, i.e. $\phi_r \approx p_c$.

A connection expires with the probability p_{expire} if the server is not saturated when the *connection arrived* event is generated. The connection expire probability can also be approximated with the steady-state probabilities as follows:

$$\phi_e \approx \sum_{k=0}^{c-1} p_k p_{\text{expire}} \tag{7}$$

In this model, the resources that the attacker spends to achieve a negative impact on the service level are proportional to the residual malicious connections arrival rate λ_m; the actual malicious traffic arrival rate at the upstream defences might be significantly higher. The resources that the server spends to achieve the required service level is represented by the capacity c of the queue. We are interested in how the tradeoff between the attacker and server resources varies for the same legitimate connection fail probability ϕ, or equivalently for the same *connection complete probability* $1-\phi$. Even though the fully expanded expression of ϕ is quite complex, what lies beneath it is a tradeoff between these quantities

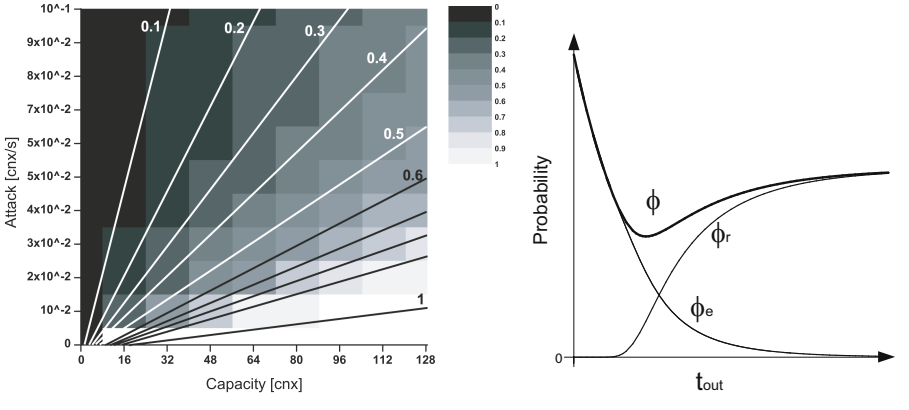


Fig. 1. (a) Steady-state legitimate connection complete probabilities for various queue capacities c (x-axis) and attack rates λ_m (y-axis), at fixed legitimate arrival rate $\lambda_l = 10$ cnx/s, mean service time $t_l = 1$ s, and timeout $t_{\text{out}} = 75$ s. For each pair (x, y), the corresponding connection complete probability is indicated as a gray-scale value for the corresponding rectangular region of the graph. Better quality of service (i.e. higher probability, lighter shades) are achieved with bigger queues and lower attack rates. The contour curves connect points (x, y) with the same connection complete probabilities (same colour), and are approximately represented by straight lines in the figure. (b) Variation of the steady-state reject, expire and fail probabilities, ϕ_r, ϕ_e, and ϕ, respectively, as a function of the timeout value t_{out}.

that is essentially linear for the same connection complete probability, as we have verified with several numerical calculations. Fig. 1(a) illustrates the contour curves for the connection complete probability, for different values of attack rates and server capacities. Note that they are essentially straight, indicating that an increase in attack rate by the attacker can be efficiently matched by a corresponding linear increase in queue capacity by the defender, while keeping the same quality of service; this confirms previously known intuition by experts in the network security field.

Although the residual traffic rates represented might seem ridiculously small, this traffic would have already been severely filtered by other upstream defences, if such were present. Thus, in order to get this small amount of residual traffic through, the attacker might have had to generate large amounts of traffic at the perimeter, resulting in a high resource cost. See Table 1 in the Appendix for default configuration parameters of different implementations of connection-oriented protocols.

Given a certain attack rate λ_m and server capacity c, the parameter that can be optimised by the defender is the timeout. As Fig. 1(b) shows, the two components of the legitimate connections reject probability ϕ, ϕ_e and ϕ_r, change in opposite directions as we vary the timeout: ϕ_e decreases exponentially with the timeout, while ϕ_r increases. When no attack is present ϕ_r is null for $\lambda_l < \mu_l c$; it has the limit $\lambda_l - \mu_l c$ for infinite timeout when $\lambda_l > \mu_l c$. When an attack is present, ϕ_r has the limit λ_l when timeout is infinite. For a specific attack rate and capacity there is an optimal timeout value that can be calculated numerically.

3 Dynamic Timeout Management Strategies

We will now analyse two queue management strategies that consist in dynamically adjusting the timeout. This is, of course, in contrast with the standard strategy of having a fixed, non-adaptive connection timeout value. Ideally we would want to make this adjustment by looking at the triangular Markov chain and choosing a timeout according to the number of legitimate and malicious connections in the server. Unfortunately, this model is not visible because the server is unable to distinguish if a connection request is legitimate or malicious. Therefore, the only information available to adjust the timeout is the total number of connections used. While the threshold prevention strategy is already implemented in Microsoft Windows Server 2003 and some security appliances, the second strategy, linear timeout prevention, is a concept that we introduce. Fig. 7 in the appendix illustrates how they fit in the taxonomy of DDoS defence of [20].

There are for each of these strategies, two alternate methods for deciding how to flush out timed out connections: *deterministic* and *deferred*. The *deterministic method* consists in tagging each connection with a pre-determined expiry time upon its arrival. The expiry time is simply the arrival time plus the timeout value at the moment of arrival. To take into account the fact that the reality of the system might have changed drastically since the arrival of a connection, another approach seems more suitable: to *defer* the assignment of an expiry time, such

that if the timeout decreases after its arrival, the connection is checked against the new timeout value. Thus at any given time, connections are flushed if the time elapsed since their arrival is bigger than the current timeout value. We refer to this method as the *deferred method*. In the rest of this section, we instantiate the general Markov models of Sect. 2 and compute steady-state probabilities for the deterministic method only. We will nonetheless present simulation results for both in Sect. 4.

3.1 Threshold-Based Timeout Adjustment Strategy

This consists in using a normal, long timeout t_0 at first. If the number of connections used in the server is greater then a certain threshold S, a shorter, attack timeout t_1 will be used. The timeout used will depend at all times on the state k in which the server is:

$$t_{\text{out}}^{(k)} = \begin{cases} t_0 & k < S \\ t_1 & otherwise \end{cases} \tag{8}$$

The probability that an individual connection will expire p_{expire}, the legitimate service rate μ_l and the approximative overall service rate $\tilde{\mu}$ described in (2), (3) and (5) all become state dependent:

$$p_{\text{expire}}^{(k)} = e^{-t_{\text{out}}^{(k)}\mu_c}\ ; \qquad \mu_l^{(k)} = \frac{\mu_c}{1 - e^{-t_{\text{out}}^{(k)}\mu_c}}\ ; \qquad \tilde{\mu}^{(k)} = \frac{\mu_l^{(k)}\mu_m(\lambda_m\mu_l^{(k)} + \lambda_l\mu_m)}{\lambda_m(\mu_l^{(k)})^2 + \lambda_l\mu_m^2} \tag{9}$$

We use the same principle as before to calculate the probability that the server is in a specific state k using Erlang's loss formula:

$$p_k = \frac{1}{k!}\prod_{j=0}^{k-1}\frac{\lambda_l + \lambda_m}{\tilde{\mu}^{(j)}} \Big/ \sum_{i=0}^{c}\left(\frac{1}{i!}\prod_{j=0}^{i-1}\frac{\lambda_l + \lambda_m}{\tilde{\mu}^{(j)}}\right) \tag{10}$$

Similar to the case where no timeout adjustment is made, the significant performance measure ϕ representing the legitimate *connection fail* event probability is calculated as:

$$\phi = \phi_r + \phi_e = p_c + \sum_{k=0}^{c-1} p_k p_{\text{expire}}^{(k)} \tag{11}$$

The tradeoff between the attacker and server resources is still linear but more favourable for the server than with a fixed timeout. Fig. 2 illustrates this tradeoff for numerical values of the rates (λ_l and μ_l), timeouts (t_0 and t_1) and threshold S similar to what we can find in Microsoft and McAfee products that use this strategy in a real-life scenario.

3.2 Linear Timeout Adjustment Strategy

This strategy differs from the threshold-based one in the way the timeout is decreased. Instead of suddenly decreasing the timeout when the server state

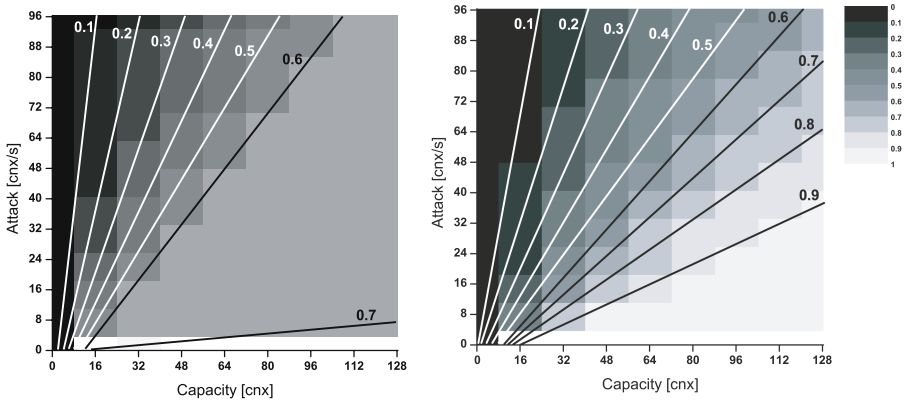


Fig. 2. Steady-state legitimate connection complete probabilities for various queue capacities c (x-axis) and attack rates λ_m (y-axis), at fixed legitimate arrival rate $\lambda_l = 10$ cnx/s, mean service time $t_l = 1$ s, and timeout values $t_0 = 75$ s, $t_1 = 1$ s for (a) a single threshold at $S = c/2$, and (b) linear adjustment. Connection complete probabilities for each combination (x, y) of queue size and attack rate is represented by gray-scaling the corresponding rectangular region.

reaches a certain threshold, this strategy gradually decreases the timeout as the number of connections in the server increases. When no connection is used (i.e. the server is in the state 0) an empty-queue long timeout t_0 is used; when all connections are used (i.e. the server is in the state c), a full-queue shorter timeout t_1 is used; and otherwise, a linear interpolation of the two values is used in all other server states. Thus, (8) becomes $t_{\text{out}}^{(k)} = t_0 + (t_1 - t_0)k/c$.

The same definitions in (9) that describe the individual *connection expire* event probability $p_{\text{expire}}^{(k)}$, the legitimate service rate $\mu_l^{(k)}$ and the approximative overall service rate $\tilde{\mu}^{(k)}$ can be inserted in the Erlang loss formula (10) to calculate the legitimate *connection fail* event probability:

$$\phi = \phi_r + \phi_e = p_c + \sum_{k=0}^{c-1} p_k p_{\text{expire}}^{(k)} \tag{12}$$

Once again, we are interested in the tradeoff between the attacker and server resources. Analysis of the two protection strategies show that for the same values of systems parameters and traffic (within the range explored), the linear timeout protection strategy could perform better than the threshold timeout protection strategy. These results are illustrated in Fig. 2(b). Finally, it is important to note that while the linear timeout adjustment strategy is slightly more complex than the threshold-based one, the computational overhead for a server implementing it is negligible.

4 Experimental Results and Interpretation

We implemented these two strategies, in both their deterministic and deferred variants, and measured their performance using a home-made traffic simulator. We also implemented and measured the performance of the standard fixed-timeout strategy, for comparison. The legitimate and residual malicious *connection requests* were generated using Poisson processes. The *connection complete* events for legitimate connections were also generated using a Poisson process. The residual attack traffic was generated in two different ways: a) a Poisson process, in order to validate the theoretical model, and b) a deterministic process with bursts of instantaneous traffic at regular time intervals; the volume of each burst adjusted such that the averaged traffic rate would always remain the same.

In the first case, we conducted simulations with parameters equivalent to those of a hardened Web server under attack. The queue capacity was set to a more realistic 8000 cnx and shorter timeout values were used: $t_{\text{out}} = 10$ s, $t_0 = 10$ s, $t_1 = 0.2$ s. The range of attack rates explored went from a modest 128 cnx/s to a very respectable 65536 cnx/s, equivalent to a 26 Mbps (!) residual attack bandwidth. For all strategies, nine different input data sets were used (except for the linear deferred, where only one simulation was run). The averaged results are shown in Fig. 3 and they give a clear picture of the relative performance of the various methods we have discussed here; the maximum standard deviation for performance in all runs was 0.023.

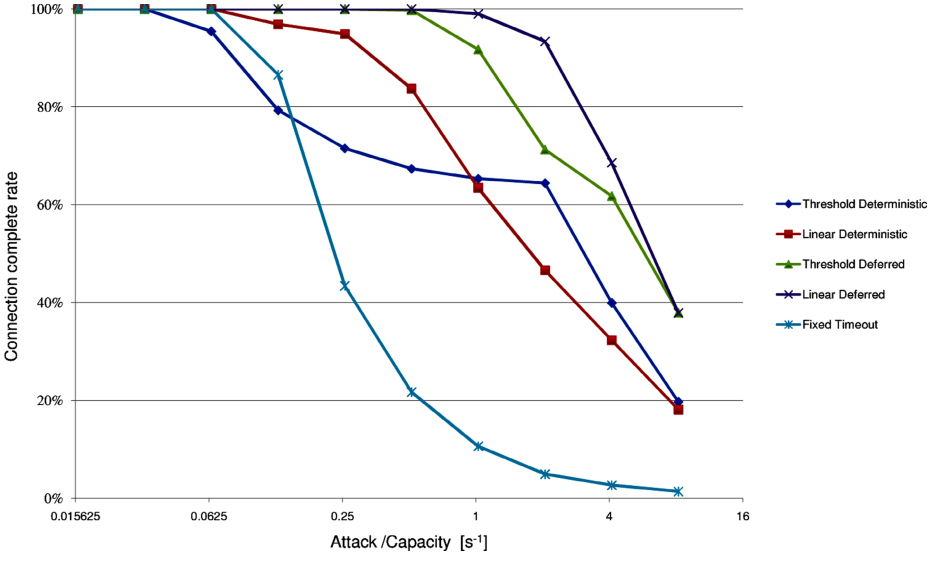


Fig. 3. Legitimate connection complete rate (y-axis) for various strategies, with fixed queue size $c = 8000$, legitimate traffic rate λ_l=100 cnx/s, mean service time $t_l = 0.2$ s, and timeout values t_0=10 s and t_1=0.2 s, for various relative virulence (x-axis)

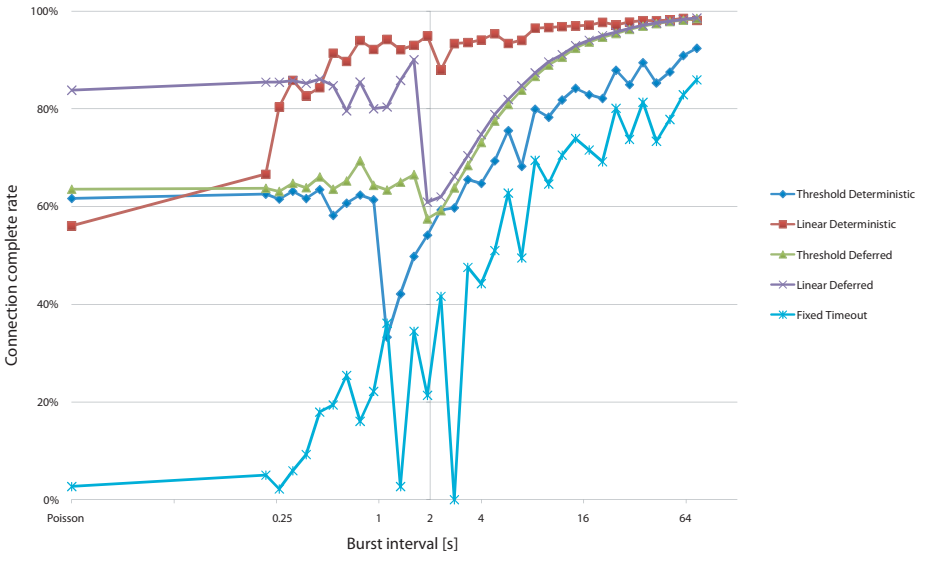


Fig. 4. Legitimate connection complete rate (y-axis) for various strategies, with fixed queue size $c = 128$ cnx, legitimate traffic rate λ_l=10 cnx/s, mean service time $t_l = 1$ s, timeout values t_0=75 s and t_1=1 s, and attack rate $\lambda_m = 64$ cnx/s, for Poisson attacks (far left) and various burst inter-arrival times (x-axis)

In order to better understand these results, it is useful to define the notion of *relative attack virulence* as the ratio between the rate of attack λ_m and the queue size c. Intuitively, it corresponds to how many queues per second the attack could fill up, if there was no timeout and no legitimate traffic. In fact, our first observation is that virulence is indeed the most important parameter affecting completion probabilities. We have confirmed this by running simulations at various combinations of attack rate and queue size, and have observed the same linearity between them as we have described in Sect. 3 for the theoretical model (see Fig. 4 in the appendix for more details).

As can be seen, at low virulence ($< 0.05\ \mathrm{s}^{-1}$) the QoS degradation is negligible, and at very high virulence ($> 16\ \mathrm{s}^{-1}$) the degradation is equally unacceptable for all strategies. In between these values, which constitutes the "window of interest" of these results, several conclusions can be drawn with respect to the relative performance of these strategies that confirm the theoretical predictions of Sect. 3. First, both timeout adjustment strategies are much better than those with a fixed timeout. Second, linear adjustment performs slightly better than the threshold-based timeout adjustment. In particular, the differences in performance can be as high as 20%, for virulence around 2 s. This corresponds to a relatively high residual attack rate of 16,000 cnx/s (6.5 Mbps) at which all strategies would notice a significant decrease in QoS (at least 30% legitimate connections lost), except the linear deferred strategy where QoS degradation would be very small (a few percent). Finally, let us emphasise that these conclusions are quite general.

We ran a separate set of simulations with values typical of an unprotected TCP stack in an unhardened OS. For the same relative virulence, the QoS degradation results obtained are very similar, hence re-confirming the relative performance of the various strategies.

In the second case, we explored the performance of these strategies against attack traffic not generated according to a Poisson process, something we could not do with our theoretical model. The results of these simulations are shown in Fig. 4, where we show the performance of the strategies for a fixed attack rate and various burst inter-arrival times. First, we notice that a Poisson attack strategy is not always optimal for the attacker, as a significant degradation of QoS happens at an inter-arrival rate of 2 s (identified with a vertical line in Fig. 4). This value is particularly significant as at this virulence level the queue is completely filled with attack traffic at every burst, and the only time that legitimate traffic can be serviced is after some of these packets have timed out and before the next burst. This is akin to a "resonance effect" where the attack characteristics are matched to those of the queue. This is optimal to the attacker, first because higher inter-arrival times results in bursts that are oversized and waste attack packets, and in addition result in an increased time window in which legitimate packets can be serviced. Consequently, QoS levels re-establish themselves linearly with respect to inter-arrival times. Second, if inter-arrival time is decreased, burst volume also decreases thus leaving space in the queue for legitimate requests arriving before the next burst to be serviced.

Nonetheless, the relative performance of the queue management strategies is the same as in the Poisson attack case. The only notable deviation is that the linear deterministic adjustment strategy is more robust to the queue resonance effect described above. Its performance is better than the linear deferred method (and all others) at all inter-arrival time settings, except for low-volume, frequent bursts.

5 Conclusion and Future Work

In this paper we made an effort to understand the effectiveness of queue management strategies against DoS attacks. We first constructed a Markov model describing the behaviour of a server under DoS attack that tries to exhaust the available connection slots in the queue. This model has allowed us to gain intuition on the likely tradeoffs between the various parameters that characterise a system under attack (traffic and service rates, queue size, etc.). Of particular interest, but relatively unexplored, is the possibility of optimising queue management parameters such as timeout and queue capacity with the respect to an expected residual attack rate and QoS requirement. There are however a few limitations to this model that should be the object of further research. First, we have used the steady-state approximations, thus assuming equilibrium, which is not accurate in the case of high residual traffic rates. Second, we have not described in this paper the model for analysing the deferred method of policing timeout connections out of the queue.

Nonetheless, from the analysis of the model in combination with the simulation results (which include non-Poisson residual traffic distributions), several interesting conclusions can be drawn that should be of immediate application for those vendors and system administrators that are incorporating or using such types of strategies in OS and applications in host servers or in anti-DoS network appliances:

1. The tradeoff between residual attack rate and queue capacity is indeed linear for almost all strategies and scenarios. This confirms previously known empirical evidence.
2. Dynamically adjusting timeout is always a good idea, except for coarse threshold-based adjustments that are overprotective in the case of light residual attack traffic.
3. Fine-grained linear timeout adjustments always outperforms fixed timeout and threshold-based adjustments, and is *significantly* better for moderate attack traffic rates.
4. The deterministic method of policing connections out of the queue is more robust to attack parameter optimisation (the "resonance effect") and has lower CPU overhead. However, the deferred method performs better against Poisson attacks, at the cost of a CPU overhead linear in the size of the queue.

We hope to further confirm these findings in future work by a) exploring a wider range of attack strategies and queue management algorithms and parameters in simulation, and b) conducting actual experiments in laboratory networks pitting various attacks against implementations of these strategies in different OS and applications. In these experiments we hope to test in conditions beyond some of the modelling assumptions made, such as Poission service rates for legimitate connections. In particular, we are aware that RTT distributions tend to be heavy-tailed [26], and we hope to test our results such conditions which are probably more realistic for normal network conditions.

Finally, while the work shown here is only applicable *as-is* to SYN-flood attacks it has the potential to be applied to other types of connection depletion attacks for TCP or other higher level protocols. One of the immediate difficulties of generalising this work, is that the standards for most relevant protocols (e.g. HTTP v1.1 [9], TLS v1.1 [6] and FTP [24]) do not define connection timeout mechanisms. Nonetheless, several applications that implement these protocols do include such timeout mechanisms (see Table 1 in the Appendix), and as such some of the results obtained might be applied to make them more resilient to the corresponding version of connection depletion attacks. Verifying this intuition for such protocol implementations is the object of ongoing research by our group.

References

1. Baras, J.: Modeling and simulation of telecommunication networks for control and management. In: Proc. Winter Simulation Conf. (2003)
2. Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A.D., Sklower, K.: Experience with DETER: A testbed for security research. In: Proc. Int. Conf. on Testbeds & Research Infrastructures for the DEvelopment of NeTworks & COMmunities (TRIDENTCOM 2006) (2006)

3. Bernstein, D.: SYN cookies (2003), http://cr.yp.to/syncookies.html
4. Cao, J., Cleveland, W., Lin, D., Sun, D.: Internet traffic tends toward Poisson and independent as the load increases. In: Denison, D., Hansen, M., Holmes, C., Mallick, B., Yu, B. (eds.) Nonlinear estimation and Classification. LNCS, vol. 171, pp. 83–110. Springer, Heidelberg (2003)
5. Cheng, C.-M., Kung, H., Tan, K.-S.: Use of spectral analysis in defense against DoS attacks. In: Proc. IEEE Global Telecommunications Conf (GLOBECOM), pp. 2143–2148. IEEE Computer Society Press, Los Alamitos (2002)
6. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol. Version 1.1. RFC 4346 (April 2006), http://tools.ietf.org/html/rfc4346
7. Feng, W., Kaiser, E., Luu, A.: Design and implementation of network puzzles. In: Proc. Annual Joint Conf. of IEEE Computer and Communications Societies (INFOCOM), vol. 4, pp. 2372–2382. IEEE Computer Society Press, Los Alamitos (2005)
8. Ferguson, P., Senie, D.: Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing RFC 2267 (January 1998), http://tools.ietf.org/html/rfc2267
9. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol – HTTP/1.1. RFC 2616 (June 1999), http://tools.ietf.org/html/rfc2616#section-8
10. Gong, F.: Deciphering detection techniques: Part III denial of service detection. McAfee Network Security Technologies Group (January 2003), http://www.mcafee.com/us/local_content/white_papers/wp_ddt_dos.pdf
11. Hoang, X., Hu, J.: An efficient hidden Markov model training scheme for anomaly intrusion detection of server applications based on system calls. In: Proc. IEEE Int. Conf. on Networks (ICON), vol. 2, pp. 470–474. IEEE Computer Society Press, Los Alamitos (2004)
12. Juels, A., Brainard, J.: Client puzzles: A cryptographic defense against connection depletion. In: Proc. Network and Distributed System Security Symposium (NDSS) (1999)
13. Khan, S., Traoré, I.: Queue-based analysis of DoS attacks. In: Proc. IEEE Work. on Information Assurance and Security (WIAS), pp. 266–273. IEEE Computer Society Press, Los Alamitos (2005)
14. Lui, J.C., Misra, V., Rubenstein, D.: On the robustness of soft state protocols. In: Proc. IEEE Int. Conf. on Network Protocols (ICNP), pp. 50–60. IEEE Computer Society Press, Los Alamitos (2004)
15. Madan, B., Goseva-Popstojanova, K., Vaidyanathan, K., Trivedi, K.: Modeling and quantification of security attributes of software systems. In: Proc. Int. Conf. on Dependable Systems and Networks (DSN), pp. 505–514 (2002)
16. Meadows, C.: A formal framework and evaluation method for network denial of service. In: Proc. IEEE Computer Security Foundations Work, IEEE Computer Society Press, Los Alamitos (1999)
17. Meadows, C.: A cost-based framework for analysis of denial of service networks. Journal of Computer Security 9(1/2), 143–164 (2001)
18. Microsoft Corporation. Security considerations for network attacks, http://www.microsoft.com/technet/security/topics/networksecurity/secdeny.mspx
19. Mirkovic, J., Dietrich, S., Dittrich, D., Reiher, P.: Internet Denial of Service: Attack and Defense Mechanisms. Prentice-Hall, Englewood Cliffs (2004)

20. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev. 34(2), 39–53 (2004)
21. Mirkovic, J., Reiher, P., Fahmy, S., Thomas, R., Hussain, A., Schwab, S., Ko, C.: Measuring denial of service. In: Proc. ACM Work. on Quality of Protection (QoP), pp. 53–58. ACM Press, New York (2006)
22. Mirkovic, J., Robinson, M., Reiher, P.: Alliance formation for DDoS defense. In: Proc. New Security Paradigms Work (NSPW), pp. 11–18. ACM SIGSAC (2003)
23. Nuzman, C., Saniee, I., Sweldens, W., Weiss, A.: A compound model for TCP connection arrivals for LAN and WAN applications. Comput. Networks 40(3), 319–337 (2002)
24. Postel, J., Reynolds, J.: File transfer protocol (FTP). RFC 959 (October 1985), http://tools.ietf.org/html/rfc959
25. Robinson, M., Mirkovic, J., Michel, S., Schnaider, M., Reiher, P.: DefCOM: defensive cooperative overlay mesh. In: Proc. DARPA Information Survivability Conf. and Exposition, vol. 2, pp. 101–102 (2003)
26. Shakkottai, S., Srikant, R., Brownlee, N., Broido, A., Claffy, K.: The RTT distribution of TCP flows in the Internet and its impact on TCP-based flow control. Technical report, Cooperative Association for Internet Data Analysis (CAIDA) (February 2004)
27. Varanasi, R., Phoha, V., Joshi, S.: IP-traceback based attacker tracking: A probabilistic technique for detecting Internet attacks using the concept of hidden markov models. In: Proc. IEEE Information Assurance Work, IEEE Computer Society Press, Los Alamitos (2004)
28. Zuquete, A.: Improving the functionality of SYN cookies. In: Proc. IFIP TC6/TC11 Joint Working Conf. on Communications and Multimedia Security, pp. 57–77 (2002)

A Additional Tables and Figures

Mean results of the legitimate connection completion rates when using the fixed-threshold and the linear timeout protection strategies are presented in Fig. 5. The standard deviation was smaller than 10^{-2} for all scenarios and strategies tested.

Table 1. Minimal attack rate exhausting all the connections of a server configured by default

Protocol	Server	Queue size c [cnx]	Timeout t_{out} [s]	Attack rate λ_m [cnx/s]
TCP	Linux 2.6.20	1024	180	5.7
	Solaris 9	1024	60	17.1
	Windows 2003	1000	21	47.6
HTTP/1.1	Apache 2.0	150	300	0.5
	IIS 6.0	8000	120	66.7

Table 2. Simulation results showing connection success rate for all strategies and different legitimate connection request rates, λ_m=10000 cnx/s, mean service time t_l=0.2 s, and timeout values t_0=10 s and t_1=0.2 s

Legit. rate λ_l [cnx/s]	No protection	Threshold det.	Threshold def.	Linear det.	Linear def
128	9.08%	66.22%	85.86%	59.63%	92.40%
16384	9.01%	64.08%	62.43%	64.93%	87.25%

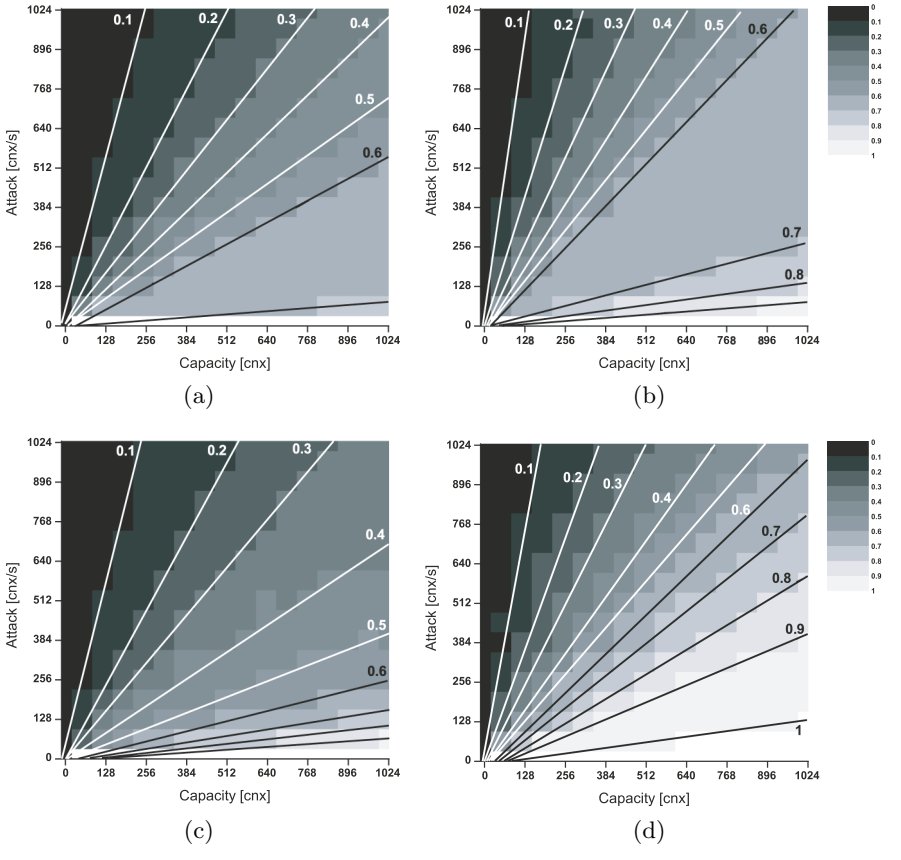


Fig. 5. Simulation results showing legitimate connection complete frequencies for various queue capacities and attack rates, λ_l=10 cnx/s, μ_l=1 cnx/s, t_0=75 s and t_1 = 1 s, for the single threshold, (a) and (b), and linear strategies, (c) and (d), using the deterministic and deferred methods, respectively

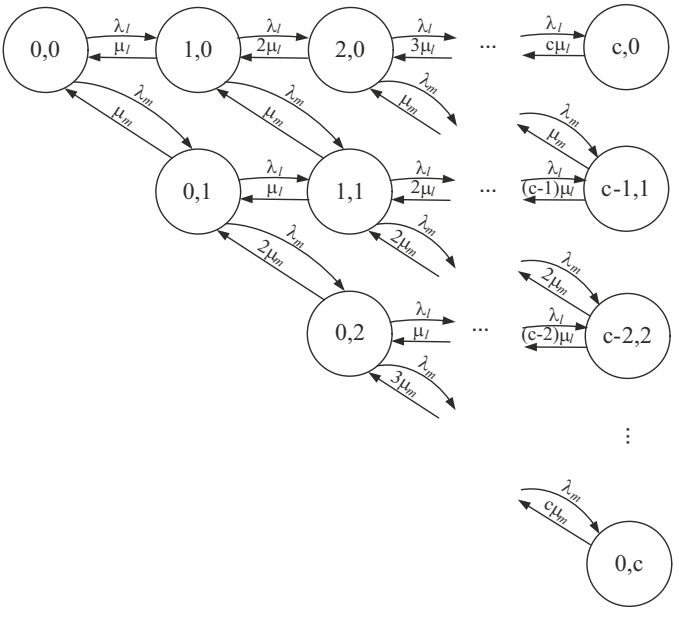


Fig. 6. Triangular DoS Markov chain model

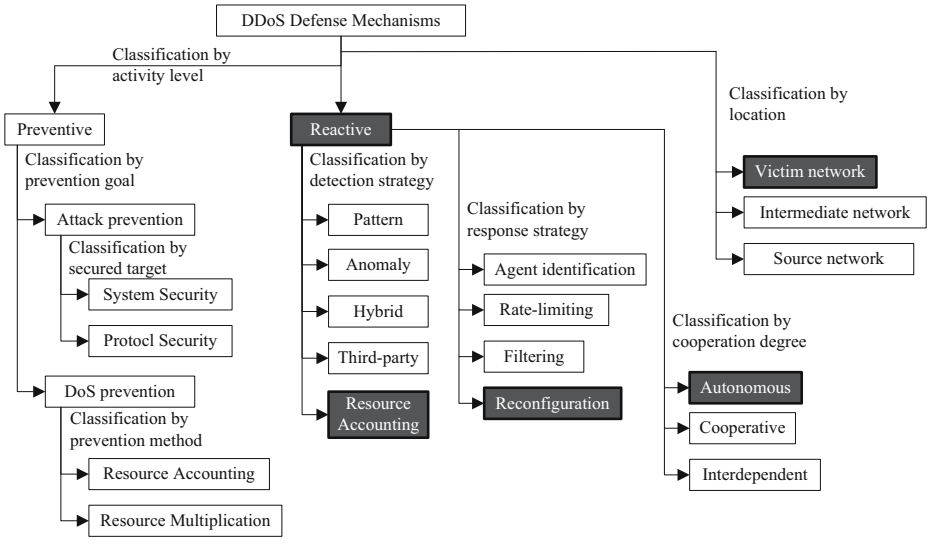


Fig. 7. Taxonomy of distributed denial-of-service defence mechanisms. The properties of the two protection strategies we analyse are highlighted.

On the Concept of Software Obfuscation in Computer Security*

Nikolay Kuzurin[1], Alexander Shokurov[1],
Nikolay Varnovsky[2], and Vladimir Zakharov[2]

[1] Institute for System Programming, Moscow, Russia
[2] Lomonosov Moscow State University, Russia

Abstract. Program obfuscation is a semantic-preserving transformation aimed at bringing a program into such a form, which impedes the understanding of its algorithm and data structures or prevents extracting of some valuable information from the text of a program. Since obfuscation could find wide use in computer security, information hiding and cryptography, security requirements to program obfuscators became a major focus of interests for pioneers of theory of software obfuscation. In this paper we also address the issue of defining security of program obfuscation. We argue that requirements to obfuscation may be different and dependent on potential applications. Therefore, it makes sense to deal with a broad spectrum of security definitions for program obfuscation. In this paper we analyze five models for studying various aspects of obfuscation: "black box" model of total obfuscation, "grey box" model of total obfuscation, obfuscation for software protection, constant hiding, and predicate obfuscation. For each of these models we consider the applications where the model may be valid, positive and negative results on the existence of secure obfuscation in the framework of the model, and relationships with other models of program obfuscation.

Keywords: Program obfuscation, security, Turing machine, encryption.

1 Introduction

To obfuscate a program means to bring it into such a form which hampers as much as possible the extraction of some valuable information concerning algorithms, data structures, secrete keys, etc. from the text of a program. Obfuscation can be viewed as a special case of encryption. One minor difference is that plaintext (original program) needs not be efficiently extractable from cryptogram (obfuscated program). The main difference seems to be decisive: a cryptogram itself must be an executable code equivalent to the original program. The latter is the reason why obfuscation attracts considerable interest of many researchers in cryptography and computer security.

Obfuscation is a relatively new topic in computer science. It was first mentioned (without using the term "obfuscation") in the seminal paper by Diffie and

* This work is supported by RFBR grant 06-01-00584.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 281–298, 2007.

Hellman [15]. When introducing the concept of public-key cryptosystem, they noticed that, given any means for obscuring data structures in a private-key encryption scheme, one could convert this algorithm into a public-key encryption scheme. Software engineering community became acquainted with obfuscation through the paper by Collberg et al. [10] where it was presented as an effective means for protecting software intellectual property against reverse engineering. These papers marked the beginning of two lines of research in program obfuscation, namely, obfuscation for the purposes of cryptography and obfuscation for software engineering and computer security.

Program obfuscators could enjoy wide application in cryptography; a secure program obfuscation would allow one to convert private-key encryption schemes into public-key ones, to design homomorphic public-key cryptosystems, to remove random oracles from cryptographic protocols, etc. But in all these cases program obfuscation should comply with some security requirements used in cryptography. Barak et al. initiated in [3] a theoretical investigation of obfuscation. They introduced the concept of "virtual black box" security: an obfuscation $\mathcal{O}$ is secure iff anything one can efficiently compute given an obfuscated program $\mathcal{O}(M)$ one could also efficiently compute given only oracle access to the original program M. They also proved that some families of functions $\mathcal{F}$ are inherently unobfuscatable in the following sense: there is a property P such that the value $P(f)$ can be efficiently computed, given any program that computes a function $f \in \mathcal{F}$, but no efficient algorithm can compute $P(f)$ when it is given only oracle access to f. This negative result was strengthened in [18,22,32] for some other variants of "virtual black box" security. Positive results were obtained also: a secure obfuscation is possible (under some rather strong cryptographic assumptions) for programs that compute point functions evaluating to zero almost everywhere (see [22,24,29,32]).

There are also a lot of problems in software engineering and computer security where program obfuscation would be very much helpful. Over the years it was found that obfuscation as a low cost means may be used for preventing reverse engineering [10,11], defending against computer viruses [9], protecting software watermarks and fingerprints [2,12], providing security of mobile agents [14,21], maintaining private searching on streaming data [26]. Unfortunately, obfuscation is also well-suited for some malevolent purposes, e.g. for obscuring malwares [7,27] and some types artificial vulnerabilities in protection systems [4]. These and some other applications have given impetus to the development of numerous obfuscation techniques (see, e.g. [1,8,11,13,23,25,31]). For the most part these approaches are no more than heuristics (sometime rather sophisticated) intended for distracting program analysis algorithms and impeding thus the understanding and reverse engineering of obfuscated programs. The principle drawback of all known obfuscation techniques is that they do not refer to any formal definition of obfuscation security and do not have a firm ground for estimating to what extent such methods serve the purpose.

This brief overview of the current state of art in software obfuscation allows some conclusions.

- The applications of program obfuscation are notably diversified, and the objectives to be pursued in these applications are also different. When obfuscation is intended to turn a private-key cryptosystem into a public-key one it is expected that such transformation hides a secret key but not necessarily an implementation of the encryption algorithm. On the contrary, when obfuscation is designed for software protection against reverse engineering it is often bound to hide only the implementation of the most valuable algorithms but not a processed data. This means that the term "obfuscation problem" is but a generic name for the set of particular problems, each being determined by the specific security requirement.
- There is considerable gap between negative and positive results. Actually, the authors of [3] proved the impossibility of a universal obfuscating program by presenting a family of functions such that every program which computes any of these functions readily betrays a secret when being applied to itself (cf. self-applicability problem for Turing machines). The positive results were obtained in [5,24,32] only for a very small class of functions under strong cryptographic assumptions. Although these results were extended in [16] to the more interesting case of proximity queries, nothing is known about the (im)possibility of effective obfuscation for common cryptographic algorithms, or any meaningful class of programs (say, finite automata) under standard cryptographic assumptions against reasonably weak security requirements.
- There is considerable gap between theory and practice of program obfuscation. There are many papers which suggest various approaches to designing obfuscating transformations of programs; some of them are implemented in academic or commercial toolkits for program obfuscation. But the influence of fundamental results from [3,18,24,32] on this branch of software engineering is minor: security requirements studied in the context of cryptographic applications are either too strong or inadequate to many software protection problems emerged in practice.

We believe that further progress in the study of program obfuscation may ensue primarily from the development of a solid framework which makes it possible to set up security definitions for program obfuscation in the context of various applications. This would help us to reveal the most important common properties required of any type of program obfuscation. Having access to a variety of security definitions one may also understand better what security requirements one or other obfuscating transformation complies with and thus estimate both the potency and the drawbacks of a particular program obfuscation technique. Intuition suggests that some weakly secure program obfuscation is possible: everybody dealing with program understanding knows that in many cases even small programs require considerable efforts to reveal their meaning. A variety of new formal security requirements for program obfuscation would offer a clearer view of this observation. Finally, when new formalizations of security requirements for software obfuscation are brought into service, this opens new

channels for adopting formal methods from computer science and cryptography to the problems of software protection.

In this paper we address the issue of defining security of program obfuscation. We argue that requirements to obfuscated programs may be different and are dependent on potential applications. To this end we consider five models for studying various aspects of obfuscation. We take a simulation paradigm and "black box" model of total obfuscation defined in [3] as a basis. Strong impossibility results obtained in [3] leave open the following question: whether there exist some weaker meaningful forms of security requirements that admits the existence of provably secure obfuscator. In attempt to find an answer to this question we try to adopt this model to various applications where program obfuscation may be used. This bring us to four new formal models of program obfuscation, namely, "grey box" model of total obfuscation, obfuscation for software protection, constant hiding, and predicate obfuscation. We study the basic properties of these models, positive and negative results on the existence of secure obfuscation in the framework of the models, and relationships with other models of program obfuscation. The key notions of program obfuscation studied in this paper have been introduced in preliminary form in Technical Report [30]. We believe that these new models of program obfuscation would provide a good framework for analyzing the potency of obfuscating transformations developed in software engineering for the purpose of program protection.

2 Notation

In complexity theory there is a number of definitions of a program. Two of them can be regarded as commonly accepted ones. The first says that program is a Turing machine, whereas the second defines program as a Boolean circuit. Accordingly, a secure obfuscation can be defined either in terms of Turing machines, or in terms of circuits (see [3]). For the sake of uniformity we restrict our consideration to Turing machines only.

We use TM as a shorthand for Turing machine. PPT denotes probabilistic polynomial-time Turing machine. For PPT A and any input x the output $A(x)$ is a random variable. When we write "for any $A(x)$" we mean a universal quantifier over the support of $A(x)$.

For a TM we will write $|A|$ to denote the size of A. For a pair of TMs A and B, $A \approx B$ denotes their equivalence, i.e. $A(x) = B(x)$ holds for any input x.

Function $\nu : N \to [0, 1]$ is negligible if it decreases faster than any inverse polynomial, i.e. for any $k \in N$ there exists n_0 such that $\nu(n) < 1/n^k$ holds for all $n \geq n_0$. We will sometimes write $neg(\cdot)$ and $poly(\cdot)$ to denote unspecified negligible function and positive polynomial, respectively.

We write $x \in_R D$ to indicate that x is chosen uniformly at random in the set D.

3 Total Obfuscation

3.1 "Black Box" Model

This concept of obfuscation was introduced and investigated by Barak et al. in the paper [3]. Ideally, a totally obfuscated program should be a "virtual black box", in the sense that anything one can deduce from its text could be also computed from its input-output behavior. This notion admits various formalizations according to what an adversary party regards as a success. The weakest form is that when an adversary tries to compute any predicate. Below we recall briefly basic definitions of "virtual black box" obfuscation security.

Definition 1 ([3]). *A probabilistic algorithm $\mathcal{O}$ is a* "virtual black box" obfuscator *if it complies with the following requirements:*

1. (functionality) *For every TM M any output $\mathcal{O}(M)$ of $\mathcal{O}$ on input M describes a TM that computes the same function as M, i.e. $M \approx \mathcal{O}(M)$.*
2. (polynomial slow-down) *The description length (size) and running time of any TM $\mathcal{O}(M)$ are at most polynomially larger than that of M, i.e. there exists a polynomial p such that for every TM M and any $\mathcal{O}(M)$, $|\mathcal{O}(M)| \leq p(|M|)$ and if M halts in t steps on some input x then $\mathcal{O}(M)$ halts within $p(t)$ steps on x.*
3. (security) *For any PPT A (adversary) there is a PPT S (simulator) and a negligible function ν such that*

$$|Pr\{A(\mathcal{O}(M)) = 1\} - Pr\{S^M(1^{|M|}) = 1\}| = \nu(|M|). \tag{1}$$

holds for all TMs M.

The main result of Barak et al. [3] is negative: secure obfuscation is impossible.

Theorem 1 ([3]). *Turing machine obfuscators (in the sense of Definition 1) do not exist.*

The key idea of the proof is as follows. For strings $\alpha, \beta \in \{0,1\}^n$ define the TMs $C_{\alpha,\beta}, D_{\alpha,\beta}, Z_n$. On every input x a TM $C_{\alpha,\beta}$ always terminates within $10|x|$ steps; it outputs β if $x = \alpha$ and 0^n otherwise. A TM $D_{\alpha,\beta}$ considers an input x as the code of some TM M, executes $M(\alpha)$ for $poly(n)$ steps, and outputs 1 if $M(\alpha) = \beta$ and 0 otherwise. A TM Z_n always outputs 0^n. Let $M_0 \# M_1$ denotes a TM which takes the pairs (δ, x), $\delta \in \{0,1\}$ as inputs and operates on x as M_0 if $\delta = 0$ and as M_1 if $\delta = 1$. Clearly, it is possible to distinguish efficiently $O(C_{\alpha,\beta} \# D_{\alpha,\beta})$ from $O(Z_n \# D_{\alpha,\beta})$ with probability 1. To this end, given a TM M on input, an adversary A first decomposes M into $M_0 \# M_1$ and then outputs $M_1(M_0)$. On the other hand, for every PPT S the outputs $S^{C_{\alpha,\beta} \# D_{\alpha,\beta}}(x)$ and $S^{Z_n \# D_{\alpha,\beta}}(x)$ will be the same with a high enough probability.

Remark 1. Programs that are obfuscatable in the sense of Definition 1 always exist. Clearly, if on every input x a TM M attaches its program to the output, i.e. $M(x) = (y, \text{"}M\text{"})$, then (1) holds trivially for any semantic-preserving

transformation $\mathcal{O}$. Note, that in this case we deal with a *learnable* program (in the sense of [28]). The question is whether there exists a class of unlearnable programs which admits a total obfuscation.

Even before "virtual black box" concept of obfuscation emerged in [3], Canetti [5] and Canetti et al. [6] essentially obfuscated point functions under various computational assumptions. A point funtion $I_\alpha(x)$ is evaluted to 1 if $x = \alpha$ and to 0 otherwise. In [24] it was shown that a total obfuscation is possible in the random oracle model for programs computing point and multi-point functions. The problem of total obfuscation of point functions was also studied by Wee in [32]. In this paper it was shown that an efficient obfuscation for the family of point functions with multi-bit output is possible in the standard model of computation provided that some very strong one-way permutations exist. Using the construction from [6] Dodis and Smith demonstrated in [16] how to obfuscate proximity queries where points are selected at random from a distribution with min-entropy.

For the most applications of program obfuscation in cryptography (including the converting of private-key encryption algorithms into public-key cryptosystems, the designing of homomorphic public-key cryptosystems, etc.) the "virtual black box" security paradigm is crucial. Another particularly challenging potential application is the removing of random oracles from cryptographic schemes. Note, however, that in this case obfuscation is not completely total. Adversary knows a priori that random oracle has to compute some function from, say, $\{0,1\}^{2n}$ to $\{0,1\}^n$ which looks random. Therefore, the method of constructing counterexamples to secure obfuscation suggested by Barak et al. [3] does not work in the case being considered. This method is based on asking the adversary to guess whether the function computed by obfuscated program is constant. For random oracles the answer is always "no". This means that at least when application in removing random oracles is concerned, security requirements to obfuscation are weaker and impossibility results of [3] do not rule out such an application. However, the most obvious approach to this problem, namely one based on obfuscating pseudorandom function families does not work yet (see [3]).

3.2 "Grey Box" Model

Since an adversary having access to an obfuscated program can always obtain not only input-output pairs but also the corresponding traces one could first pose a problem of whether it is possible to transform a program in such a way that these traces are essentially the only useful information available to an adversary. If this were possible then the next problem is how one can guarantee that the traces themselves give away no useful information.

To this end we modify the "virtual black box" obfuscation model. Definition of this new brand of obfuscation which from now on we will call "virtual grey box" obfuscation differs from Definition 1 in two aspects. The first is the specification of the oracle ("black-box") used by simulating machine S. When the machine S issues query x to the oracle it gets as a response not only the output word

y but also the trace of execution of M on input x. Note that in our setting it makes difference which particular program equivalent to M is used by the oracle. We insist that this is just the original program M. This means that we do not require obfuscator to hide any property that is learnable efficiently from traces of execution of TM M. Notice also that any obfuscator can be deemed secure only if it does not reveal any properties that remain hidden given access to the traces of original program.

The second aspect concerns the programs to be obfuscated. We consider so called reactive programs. Unlike ordinary programs intended for computing some input-output relation, reactive programs are intended to interact with the environment. This interaction displays itself in the responses a reactive program computes on the requests from the environment. Thus, a reactive program computes a function which maps the infinite sequences of inputs $x_1, x_2, \ldots, x_n, \ldots$ (sequences of requests) into the infinite sequences of outputs $y_1, y_2, \ldots, y_n, \ldots$ (sequences of replies) so that every output y_n depends on the inputs $x_1, x_2, \ldots, x_n$ only. Examples of reactive programs may be found in network protocols (including cryptographic ones), embedded systems, operating systems, etc. A reactive program may be formalized as a conventional TM which operates on read-only input tape and write-only output tape (and using some auxiliary tapes) in such a way that it does not proceed to read next input word x_{n+1} until it completes writing the output y_n. We will denote these machines as RTM to distinguish them from the ordinary TM. It should be noticed that every TM may be viewed as an RTM which resets its control into some predefined initial state s_0 and erases its auxiliary tapes every time before reading the next input x_n.

To distinguish the oracle used in Definition 1 from the oracle provided to simulating machine S in this new setting we denote the latter by $Tr(M)$. On input x this oracle outputs a pair $(y, tr_M(x))$, where y is the output of RTM M on input x (note that y depends not only on x but also on all previous inputs that have been used as requests to the oracle) and $tr_M(x)$ is the trace of execution of M on this input. The string tr_M is defined as concatenation of all successive instructions executed by M when running on input x.

Definition 2. *A probabilistic algorithm $\mathcal{O}$ is a* "virtual grey box" RTM obfuscator *if it complies with the following requirements:*

1. *For every RTM M any output $\mathcal{O}(M)$ of $\mathcal{O}$ on input M describes a RTM that is equivalent to M in the following sense: for any sequence of inputs $x_1, \ldots, x_n, \ldots$ the corresponding sequences of outputs of RTMs $\mathcal{O}(M)$ and M coincide.*
2. *The description length and running time (on every finite sequence of inputs $x_1, x_2, \ldots, x_n$) of any RTM $\mathcal{O}(M)$ are at most polynomially larger than that of M.*
3. *For any PPT A (adversary) there is a PPT S (simulator) and a negligible function ν such that*

$$|Pr\{A(\mathcal{O}(M)) = 1\} - Pr\{S^{Tr(M)}(1^{|M|}) = 1\}| = \nu(|M|). \tag{2}$$

holds for all RTMs M.

Theorem 2. *If one-way functions exist then "virtual grey box" RTM obfuscators do not exist.*

The intuition behind proof is quite simple. Barak et al. [3] proved Theorem 1 by constructing an infinite family of TMs such that certain predicate $\pi(M)$ defined on this family is unlearnable with oracle access to the function computed by M but can be decided easily given a text of any program equivalent to M. It suffices to modify this construction in such a way that the following two requirements hold simultaneously. First, learnability of predicate $\pi(M)$ given a text of any program equivalent to M should be preserved. Second, traces of execution of M must provide a simulator with no additional useful information as compared to the output-only oracle treated by Definition 1. A simulator S having access to traces of execution of M is apparently more powerful then one bounded to see input-output pairs only. To defeat this powerful simulator we make use of cryptographic tools. Namely, instead of comparing input x to the fixed string α as in Barak et al [3] we test first whether $f(x) = f(\alpha)$, where f is a one-way function. Only if this test passes we check whether $x = \alpha$.

A typical trace $tr_M(x)$ consists of instructions used to compute $f(x)$ and to check whether $f(x) = f(\alpha)$. Most of the time this check fails. Moreover, if the equality $f(x) = f(\alpha)$ holds with nonnegligible probability then this contradicts the fact that f is a one-way function.

Note that one cannot replace one-way function f in this construction by e.g. encryption function of private-key cryptosystem since encryption algorithm uses private key and traces of its execution become available to adversary.

We stress that the above discussion has to be considered as motivating. The actual construction is somewhat more complicated. Now we turn to the formal proof.

Proof. The counterexample of Barak et al. from [3] involves two families of TMs. For any pair of strings $\alpha, \beta \in \{0,1\}^n$ Turing machine $C'_{\alpha,\beta}(x)$ outputs β if $x = \alpha$ and 0^n otherwise. For the same parameters α, β Turing machine $D'_{\alpha,\beta}(C)$ outputs 1 if $C(\alpha) = \beta$ and 0 otherwise.

Let $\{f : \{0,1\}^n \to \{0,1\}^n\}_n$ be a one-way function. Our construction depends also on integer parameter $t \geq 2$ which can be e.g. a constant or a value of arbitrary but fixed polynomial (in n).

First we choose $2t$ strings $\alpha_1, \ldots, \alpha_t, \beta_1, \ldots, \beta_t \in \{0,1\}^n$ uniformly at random. Denote this $2t$-tuple by γ. Now define RTM $C_\gamma(x)$ as follows. On input x this RTM computes $f(x)$ and tests the result against precomputed values $f(\alpha_i)$, $i = 1, \ldots, t$. If $f(x) \neq f(\alpha_i)$ for all $i = 1, \ldots, t$ then C_γ outputs 0^n and halts. In the case when $f(x) = f(\alpha_i)$ for some i, C_γ checks whether $x = \alpha_i$ and if so outputs β_i, otherwise it outputs 0^n and in either case halts.

Next we define RTM D_γ. It stores two arrays of strings $\alpha_1, \ldots, \alpha_t$ and $\beta_1, \ldots, \beta_t$. Let i be a pointer to both this arrays which is initially set to 0.

On input a RTM C, D_γ runs as follows.

1. Check whether $i = t - 1$. If so, output 0 and halt.
2. Advance the pointer $i = i + 1$ and then feed C with input α_i.

3. If $C(\alpha_i) = \beta_i$ then check the current value of the pointer. If $i = t$ then output 1 and halt, else goto 2.
4. If $C(\alpha_i) \neq \beta_i$ then output 0 and halt.

Note that simulating machine having access to C_γ and D_γ as oracles may query D_γ with arbitrary RTM C and extract α_1 from the trace, then query C_γ with α_1 and obtain β_1 (even if we have a modified RTM D_γ that hides the values β_i) and so on. Therefore simulating machine is granted only $t-1$ queries to D_γ (in fact further queries are allowed but result invariably in zero responses and thus can be ignored). The unobfuscatable property is the existence of t distinct strings $\alpha_1, \ldots, \alpha_t \in \{0,1\}^n$ such that output of a given RTM on each of them is nonzero.

Pair of RTMs (C_γ, D_γ) replaces TMs $(C'_{\alpha,\beta}, D'_{\alpha,\beta})$ used in the proof of Theorem 1 presented in [3]. Analysis of this proof shows that to adopt it to our case one needs only to prove the next claim.

Claim: *Suppose that RTM Z_γ differs from C_γ only in that on input α_t it outputs 0^n. Then for any PPT S the amount*

$$|Pr\{S^{C_\gamma, D_\gamma}(1^n) = 1\} - Pr\{S^{Z_\gamma, D_\gamma}(1^n) = 1\}|$$

is negligible. The probabilities are taken over uniform choice of $\alpha_1, \ldots, \alpha_t$, $\beta_1, \ldots, \beta_t$ in $\{0,1\}^n$ and coin tosses of S.

Proof. (*Sketch*) It suffices to show that any PPT has only negligible probability to get nonzero response to any of its oracle queries, no matter which RTM, C_γ or Z_γ, is used as the first oracle.

For the sake of simplicity we assume $t = 2$ for the rest of proof. In this case machine S can issue unique query to the second oracle. Let $a_1, \ldots, a_s \in \{0,1\}^n$ be all the queries of S to the first oracle prior to issuing its only query to the second one (in fact s is a random variable). Suppose that nonzero string appears with nonnegligible probability among responses to these s queries. We construct a PPT T inverting the function f.

Let $z \in_R \{0,1\}^n$ and $y = f(z)$ be input to T. Machine T flips a coin and decides which of values, $f(\alpha_1)$ or $f(\alpha_2)$ will be set to y. Without loss of generality let it be α_1. Then T chooses $\alpha_2, \beta_1, \beta_2 \in \{0,1\}^n$ uniformly at random and calls S as a subroutine. All queries to the first oracle are intercepted by T. For a query $x \in \{0,1\}^n$, T computes $f(x)$ and checks whether $f(x) = y$ or $f(x) = f(\alpha_2)$. If neither of these equalities holds, T outputs 0^n. In the case $f(x) = y$, T outputs β_1, and in the case $f(x) = f(\alpha_2)$ it proceeds in the same way as RTM C_γ does.

The crucial observation is that the probability of seeing a nonzero response from this simulated oracle T is at least as high as in the case of real oracle. Therefore for some $j \in 1, \ldots s$ the response of T is nonzero with nonnegligible probability. For this j, α_j is in the preimage of y with probability $1/2$. Thus, T inverts f with nonnegligible probability which contradicts the fact that f is one-way function.

If nonzero string appears among responses to the first s queries with negligible probability then the probability of nonzero response to a unique query of S to the second oracle is negligible as well.

For the remaining oracle queries (after the query to the second oracle) the same argument as above shows that nonnegligible probability of success would imply existence of efficient algorithm for inverting the function f. This contradiction proves the claim. □

As it may be seen from the proof, we lean to a large measure upon the ability of an RTM to keep in memory a "history" of its computation on previous inputs. This makes such programs non-resettable. It is unclear yet, whether this theorem can be extended to ordinary TMs. This remains as an open question.

4 Obfuscation for Software Protection

As the most apparent application of obfuscation for software protection consider the following scenario. Suppose one party invents a fast algorithm for factoring integers and wishes to sell to another party a program for breaking the RSA cryptosystem. The goal is to have this program transformed in such a way that it will be hard to derive factorization algorithm from the text of transformed program. However, the "black box" obfuscation model underlying Definition 1 does not seem to be well-suited for this application. Indeed, it is hardly possible to find a client who will buy a black-box as a piece of software. Instead, any program product on the market must have a user guide specifying its functionality. In this setting an adversary knows the function computed by the program in question. The aim of obfuscation in this case is not to hide any property of the program which refers to its functionality (we may assume that such properties are known to an adversary party in advance from, say, users manual of the program), but to make unintelligible the implementation of these functional properties in a particular program. We think that this view of obfuscation is more adequate for the needs of software engineering community than total obfuscation. The most straightforward way to formalize this concept is to fix some program M_0 equivalent to the original program M and give M_0 to the adversary as an additional input. For example, if M is a program totally breaking the RSA cryptosystem, i.e. finding its private key given a public one, then M_0 might be an easy-to-understand program which accomplishes the same task by exhaustive search.

The simulating machine S guaranteed by Definition 1 also has access to the text of the program M_0. The modified Definition 3 is as follows.

Definition 3. *A probabilistic algorithm $\mathcal{O}$ is a* software protecting obfuscator *if it complies with* functionality *and* polynomial slow-down *requirements from Definition 1, and with the following* security *requirement:*

3) For any PPT A there is a PPT S such that

$$|Pr\{A(\mathcal{O}(M), \widetilde{M}) = 1\} - Pr\{S^M(1^{|M|}, \widetilde{M}) = 1\}| = neg(|M|). \quad (3)$$

holds for every pair of TMs $(M, \widetilde{M})$, where $M \approx \widetilde{M}$, and $|\widetilde{M}| = poly(|M|)$.

The machines A and S are polynomial in the lengths of their first inputs, $\mathcal{O}(M)$ and $1^{|M|}$ respectively. Note that in the case when algorithm $\widetilde{M}$ is efficient simulating machine S needs no access to the oracle M.

Remark 2. Some program properties (predicates) P are invariant under equivalent program transformations, i.e. $P(M) = P(M')$ holds for any program M' functionally equivalent to M. Clearly, such properties are only trivially obfuscatable. Note, however, that in the context of software protection it is useless to obfuscate invariant properties as soon as functionality of a program is known.

Security requirement in Definition 3 is much different from that in total obfuscation. The negative results of [3] do not rule out the possibility of "black box" secure obfuscation for many important general classes of machines. Consider as an example the obfuscation problem for deterministic finite state automata (DFA). Secure total obfuscation of DFAs still remains one of the most challenging problem in the theory of program obfuscation. But if one manages to estimate the obfuscation security following Definition 3 then a surprisingly simple obfuscation of DFAs can be obtained.

Theorem 3. *There exists a secure software protecting obfuscator for the family of DFAs.*

Proof. A DFA is but a one-way TM. We may regard any efficient DFA minimizing algorithm $\mathcal{O}$ as a TM obfuscator. Indeed, it is well known that for every DFA M there exists the unique minimal DFA M_0 such that $M \approx M_0$. Therefore, a simulator S, given a TM $\widetilde{M}$ only, can readily compute $\mathcal{O}(M)$ by applying any efficient minimizing algorithm to $\widetilde{M}$ and then mimic an adversary A on the pair $(M, \widetilde{M})$ to satisfy (3). □

The possibility of such effect has been noted in [3] and also studied in more details in [19]. It appears in software engineering as well: if a program compilation includes an advanced optimization (minimization) then executables become far less intelligible by means of reverse engineering. As it can be seen from Definition 3, software protecting obfuscators are intended to remove all individual specific features from a program to be protected. Hence, to obfuscate a program it is sufficient to reduce it to some normal form. If a family of programs $\mathcal{H}$ admits strong and effective normalization, i. e. there exists an algorithm which reduces any two equivalent programs to the same normal form, then Theorem 3 can be extended to this family: any efficient normalization algorithm becomes an obfuscator $\mathcal{O}$ for programs from $\mathcal{H}$. It is worth noting that the authors of [31] implemented a sort of normalization, namely, flattening of program control flow, as the basic obfuscating transformation in their software protection toolkit.

Unfortunately, efficient normalization is possible only for a few natural classes of programs. Therefore, we qualify TM obfuscation through normalization as trivial and pose the following question:

Open problem. Is there any class of programs which admits non-trivial secure obfuscation in the sense of Definition 3?

Another distinctive feature of TM obfuscation is that impossibility results of Barak et al. [3] do not apply in this setting. Moreover, if one attempts to extend the proof technique of the paper [3] to show that secure obfuscation in the sense of Definition 3 is impossible, then this will immediately imply that some weak form of obfuscation does exist! Indeed, counterexample of Barak et al. is based on a specific invariant property of a particular family of programs. Suppose there exists an infinite sequence of pairs $(M, \widetilde{M})$ of TMs such that some invariant property π is learnable efficiently given the text of the program $\mathcal{O}(M)$ but is unlearnable when we have the text of the program $\widetilde{M}$ and black-box access to M. But this means that $\widetilde{M}$ is a (provably) secure obfuscation of $\mathcal{O}(M)$ with respect to the property π.

It makes sense to juxtapose the notion of TM obfuscation (Definition 3) with the notion of obfuscation w.r.t. auxiliary inputs introduced in the paper [18]. Goldwasser and Kalai modified the "virtual black box" property by admitting both an adversary A and a simulator S an access to an auxiliary input z which may be dependent as well as independent from a program to be obfuscated. When being adapted to TM notation it is as follows.

Definition 4 ([18]). *A probabilistic algorithm $\mathcal{O}$ is an* obfuscator w.r.t. dependent auxiliary input *for the family of TM $\mathcal{F}$ if it complies with* functionality *and* polynomial slow-down *requirements from Definition 1, and with the following* security *requirement:*

3) For any PPT A there is a PPT S such that

$$|Pr\{A[\mathcal{O}(M), z] = 1\} - Pr\{S^M[1^{|M|}, z] = 1\} = neg(|M|)$$

holds for every TM $M \in \mathcal{F}$, and every auxiliary input z of size poly$(|M|)$ (z may depend on M).

In [18] it was shown that many natural classes of functions (so called filter functions based on circuits with super-polynomial pseudo entropy on inputs from NP-complete language) cannot be obfuscated. The security requirement in the sense of Definition 3 is much weaker than that from [18] since in the case of TM obfuscation we are bound to check (3) only for those auxiliary inputs that are TMs $\widetilde{M}$ equivalent to an obfuscated TM M. Therefore, negative results and examples of [18] cannot testify the impossibility of universal software protecting obfuscation, and we may pose the following question.

Open problem. Does there exist a class of programs that are unobfuscatable in the sense of Definition 3?

Software protecting obfuscation has much in common with best possible obfuscation advanced by Goldwasser and Rothblum in [19]. An obfuscator O is judged as best possible if it transforms any program so that anything that can be computed given an access to the obfuscated program $O(M)$ should also be computable from any other equivalent program (of some related size).

Definition 5 ([19]). *A probabilistic algorithm $\mathcal{O}$ is a* best possible obfuscator *if it complies with* functionality *and* polynomial slow-down *requirements from Definition 1, and with the following* security *requirement:*

3) For any PPT A (learner) there is a PPT S (simulator) such that for every pair of equivalent TMs $(M, \widetilde{M})$ of the same large enough size the distributions $A(O(M))$ and $S(\widetilde{M})$ are indistinguishable.

The aim of software protection is to hide all specific properties of any particular implementation M of some function known to all (the latter is formalized by giving an adversery a description of this function as an arbitrary program $\widetilde{M}$ equivalent to M). Intuitively, this may be achieved iff for every pair of implementations M_1 and M_2 of the same function the distributions $O(M_1)$ and $O(M_2)$ are indistinguishable. But, as it was proved in [19], this is equivalent (in the case of efficient O) to the claim that O is a best possible obfuscator. Thus, it may be conjectured that software protection obfuscation is equivalent to best possible obfuscation. But to prove this conjecture formally one has to elaborate Definitions 3 and 5 to circumvent the discrepancy in the admissible size of an auxiliary program $\widetilde{M}$.

5 Constant Hiding

In this setting everything to be hidden from an adversary is a certain constant c_0 used by a program M. This constant is assumed to be chosen randomly in sufficiently large set C. One can safely assume that the original program M is completely known to the adversary except for the constant c_0.

It is easy to see that constant hiding is closely related with obfuscation for software protection. Therefore we adopt Definition 3 to the present case by setting $\widetilde{M}$ to be the same TM as M except for the constant c_0 replaced by a constant c chosen randomly and independently in C. Let M be a program which uses a constant c. For simplicity we assume that $c \in_R \{0,1\}^n$ for every integer n. For any given constant value $c \in \{0,1\}^n$ we denote by M_c the corresponding instantiation of the program M. Thus, we deal with a parameterized family of programs $\mathcal{F} = \{M_c \ : \ c \in \{0,1\}^n, \ n \geq 1\}$. The obfuscation of $\mathcal{F}$ is intended to prevent the extraction of information about a constant c from the program M_c.

Definition 6. *Let $\mathcal{F}$ be a parameterized family of TMs M_c. A probabilistic algorithm $\mathcal{O}$ is a* constant hiding obfuscator *for the family $\mathcal{F}$ if it complies with* functionality *and* polynomial slow-down *requirements from Definition 1, and with the following* security *requirement:*

3) For any PPT A there is a PPT S such that

$$|Pr\{A[\mathcal{O}(M_{c_0}), M_c] = 1\} - Pr\{S^{M_{c_0}}[1^{|M_{c_0}|}, M_c] = 1\} = neg(n) \qquad (4)$$

holds for any constant value $c_0 \in \{0,1\}^n$ and $c \in_R \{0,1\}^n$.

One can define a weak form of constant hiding in which an adversary A and a simulating machine S instead of outputting a single bit have to guess the constant c_0.

Remark 3. Suppose that U is a universal TM. Then a constant c may be interpreted as an encoded description of some other TM M simulated by the instantiation U_c. In this case a constant hiding obfuscator for a universal TM is some specific variant of obfuscation w.r.t. dependent auxiliary input introduced in [18] (see Definition 4). At the same time, if all TMs M_c from the family $\mathcal{F}$ do not refer to a constant c along any execution then every semantic-preserving transformation $\mathcal{O}$ is a trivial constant hiding obfuscator for $\mathcal{F}$. Nontrivial constant hiding does exist (at least in the weak form) under cryptographic assumptions for certain restricted classes of programs. In fact, any public-key cryptosystem gives such an example since its encryption program can be regarded as hiding a particular constant, namely, a private key. The obfuscations of point functions from [24,32] can be also viewed as constant hiding obfuscations (see also [18]).

Yet another variant of constant hiding obfuscation is defined in [22]. An obfuscator $\mathcal{O}$ for a family of programs $\mathcal{F}$ is regarded as a composition of two algorithms: a key transformation routine T that takes a key c and returns an obfuscated key $c' = T(c)$, and a polynomial time machine G which on input an obfuscated key c' and x outputs $y = M_c(x)$. The authors of [22] proved that if a secure secrete-key encryption scheme can be obfuscated according to their definition, then the result is a secure public-key encryption scheme. On the other hand, they also showed that there exists a secure probabilistic secret-key cryptosystem that cannot be obfuscated. Definition 6 specifies a more general model of constant hiding obfuscation than that of [22], since in our model it is assumed that obfuscating transformations may be applied not only to a constant c_0 to be hidden but also to the program M_{c_0} which is instantiated by the constant. Instead of using some fixed universal machine G we grant an adversary an access to a typical program M_c, since all TM from the family $\mathcal{F}$ are the same except for the constants.

6 Predicate Obfuscation

This is one of the weakest model of obfuscation. It is intended to hide only a particular distinguished property of a program (predicate). Nevertheless, predicate obfuscation may appear in numerous applications in computer security. Assume that some programs are infected with a virus which is triggered off at some executions. There are several approaches to a virus detection. A sandbox approach emulates the operating system, runs executables in this simulation and analyzes the sandbox for any changes which might indicate a virus. This approach is resource consuming and normally it takes place only during on-demand scans. The most effective virus detection technique is based on virus dictionary: an antivirus program tries to find virus-patterns inside ordinary programs by scanning them for so-called *virus signatures*. To avoid detection some viruses attempt to hide their signatures by using complex obfuscating transformations and rewriting their bodies completely each time they are to infect new executables. Also the viruses tend to "weave" their code into the host program making detection by traditional heuristics almost impossible since the virus share a great deal of its instructions with a host program code. A virus-detection problem may be

specified by a predicate $\pi(M)$ which evaluates to *true* whenever the program M includes some malware piece of code. An obfuscating transformation $\mathcal{O}$ used by a metamorphic malware could be regarded secure if any anti-virus scanner is no more effective in detecting an obfuscated virus signature (i.e. in evaluating $\pi(\mathcal{O}(M))$) than some sandbox virus detector.

To formalize this idea consider some predicate P defined on a family of TMs $\mathcal{F}$.

Definition 7. *A probabilistic algorithm $\mathcal{O}$ is an* obfuscator of the predicate π *on the family $\mathcal{F}$ if it complies with* functionality *and* polynomial slow-down *requirements from Definition 1, and with the following* security *requirement:*

3) For any PPT A there is a PPT S such that

$$|Pr\{A[\mathcal{O}(M)] = \pi(M)\} - Pr\{S^M[1^{|M|}] = \pi(M)\}| = neg(|M|) \qquad (5)$$

holds for every TM M from $\mathcal{F}$ and its obfuscation $\mathcal{O}(M)$.

Predicate obfuscation is related intrinsically with total obfuscation: a "virtual black box" obfuscator needs to hide all predicates, whereas Definition 7 allows to build different obfuscators for different predicates. As it may be seen from the proof of Theorem 1 given in [3], the impossibility of total obfuscation is certified by presenting an unobfuscatable predicate as a counterexample. On the other hand, predicate obfuscation is much weaker than total obfuscation. In [29] it was shown that if every TM from a family $\mathcal{F}$ computes either zero function, or a point function, then secure obfuscation of the predicate "$M(x) \not\equiv 0$" on the family $\mathcal{F}$ is possible under assumption that a one-way permutation exists. This obfuscation is based on the *hard-core predicate* construction suggested by Goldreich and Levin [17]. As it can be seen from theorem below, predicate obfuscation offers some nice properties: composition of predicate obfuscators increases the potency of obfuscation.

We say that a predicate π on TMs is a *functional property* if $\pi(M_1) = \pi(M_2)$ holds for every pair of equivalent TMs M_1, M_2.

Theorem 4. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be obfuscator of functional properties π_1 and π_2, respectively. Suppose also that the range of O_2 is contained in the domain of O_1. Then the composition $\widehat{\mathcal{O}} = \mathcal{O}_1\mathcal{O}_2$ is an obfuscator of both predicates π_1 and π_2.*

Proof. We may assume that each obfuscator $\mathcal{O}_i$, $i = 1, 2$, is supplied with a polynomial $q_i(\cdot)$, and for every TM M the size of $\mathcal{O}_i(M)$ is exactly $q_i(|M|)$. This may be achieved by padding an obfuscated program with a sufficient amount of dummy instructions. Clearly, $\widehat{\mathcal{O}}$ satisfies the functionality and polynomial slow-down requirements.

Let M be an arbitrary TM.

1. To show that $\widehat{\mathcal{O}}$ obfuscates π_1 consider an arbitrary adversary $\widehat{A}$. Since $\mathcal{O}_1$ is an obfuscator of π_1, there exists a PPT S such that

$$|Pr\{\widehat{A}[\mathcal{O}_1(\mathcal{O}_2(M))] = \pi_1(\mathcal{O}_2(M))\} - Pr\{S^{\mathcal{O}_2(M)}[1^{|\mathcal{O}_2(M)|}] = \pi_1(\mathcal{O}_2(M))\}| = neg(|M|)$$

holds for the predicate π_1, TM $\mathcal{O}_2(M)$ and its obfuscation $\mathcal{O}_1(\mathcal{O}_2(M))$. Now we may consider a simulator $\widehat{S}$ which operates as follows: given an input 1^x, it computes $y = q_2(x)$, and applies the simulator S to 1^y. Since $\pi_1(\mathcal{O}_2(M)) = \pi_1(M)$ and $\mathcal{O}_2(M) \approx M$, we arrive at the equation

$$|Pr\{\widehat{A}[\widehat{\mathcal{O}}(M)] = \pi_1(M)\} - Pr\{\widehat{S}^M[1^{|M|}] = \pi_1(M)\}| = neg(|M|),$$

which is exactly the security requirement for π_1, TM M and its obfuscation $\widehat{\mathcal{O}}(M)$. Hence, $\widehat{\mathcal{O}}$ obfuscates π_1.

2. To show that $\widehat{\mathcal{O}}$ obfuscates π_2 consider an arbitrary adversary $\widehat{A}$ and the composition $A = \widehat{A}\mathcal{O}_1$. Since $\mathcal{O}_2$ is an obfuscator of π_2, there exists a PPT $\widehat{S}$ which simulates the adversary A so that

$$|Pr\{A[\mathcal{O}_2(M)] = \pi_2(M)\} - Pr\{S^M[1^{|M|}] = \pi_2(M)\}| = neg(|M|)$$

holds. Hence, $\widehat{\mathcal{O}}$ satisfies security requirement to predicate obfuscation for π_2. □

7 Conclusion

The models of program obfuscation presented in this paper evolved naturally from the "black box" simulation principle offered by Barak et al. in [3]. Security requirements given in Definitions 3, 6, and 7 make it possible to cover those cases of program obfuscation that do not fall into the "virtual black box" security paradigm. Although we have made only preliminary study of these new models, even such simple results as Theorems 3 and 4 show that as security requirements become weaker program obfuscation displays some new interesting features. On the other hand, Theorem 2 demonstrates that total program obfuscation remains impossible even in a framework of a weak "virtual grey box" obfuscation model under standard cryptographic assumptions.

Acknowledgements

We thank the anonymous reviewers for their insightful comments which help us to improve the paper.

References

1. Aucsmith, D.: Tamper resistant software: an implementation. In: Anderson, R. (ed.) Information Hiding. LNCS, vol. 1174, pp. 317–333. Springer, Heidelberg (1996)
2. Arboit, G.: A method for watermarking java programs via opaque predicates. 5-th International Conference on Electronic Commerce Research (2002)

3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
4. Bhatkar, S., DuVarney, D.C., Sekar, R.: Efficient techniques for comprehensive protection from memory error exploits. USENIX Security (2005)
5. Canetti, R.: Towards realizing random oracles: hash functions that hide all partial information. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997)
6. Canetti, R., Micciancio D.D., Reingold, O: Perfectly one-way probabilistic hash functions. 30-th ACM Symposium on Theory of Computing, 131–140 (1998)
7. Chess, D., White, S.: An undetectable computer virus. In: 2000 Virus Bulletin Conference (2000)
8. Chow, S., Gu, Y., Johnson, H., Zakharov, V.: An approach to obfuscation of control-flow of sequential programs. In: Wilhelm, R. (ed.) Informatics. LNCS, vol. 2000, pp. 144–155. Springer, Heidelberg (2001)
9. Cohen, F.: Operating system protection through program evolution. Computers and Security 12(6), 565–584 (1993)
10. Collberg, C., Thomborson, C., Low, D.: A Taxonomy of Obfuscating Transformations. Tech. Report, N 148, Univ. of Auckland (1997)
11. Collberg, C., Thomborson, C., Low, D., Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient and stealthy opaque constructs. In: Symposium on Principles of Programming Languages, pp. 184–196 (1998)
12. Collberg, C., Thomborson, C.: Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection. IEEE Transactions on Software Engineering 28(6) (2002)
13. Dalla Preda, M., Giacobazzi, R.: Semantic-based code obfuscation by abstract interpretation. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1325–1336. Springer, Heidelberg (2005)
14. D'Anna, L., Matt, B., Reisse, A., Van Vleck, T., Schwab, S., LeBlanc, P.: Self- Protecting Mobile Agents Obfuscation Report, Report #03-015, Network Associates Laboratories (June 2003)
15. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions om Information Theory IT-22(6), 644–654 (1976)
16. Dodis, Y., Smith, A.: Correcting errors without leaking partial informtion. 37-th ACM Symposium on Theory of Computing, 654–663 (2005)
17. Goldreich, O., Levin, L: A hard-core predicate to any one-way function. 21-th ACM Symposium on Theory of Computing, 210–217 (1989)
18. Goldwasser, S., Tauman Kalai, Y.: On the impossibility of obfuscation with auxiliary input. 46-th IEEE Symposium on Foundations of Computer Science, 553–562 (2005)
19. Goldwasser, S., Rothblum, G.N.: On best possible obfuction. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 194–213. Springer, Heidelberg (2007)
20. Hada, S.: Zero-knowledge and code obfuscation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 443–457. Springer, Heidelberg (2000)
21. Hohl, F.: Time limited blackbox security: protecting mobile agents from malicious hosts. In: Vigna, G. (ed.) Mobile Agents and Security. LNCS, vol. 1419, pp. 92–113. Springer, Heidelberg (1998)
22. Hofheinz, D., Malone-Lee, J., Stam, M.: Obfuscation for cryptographic purpose. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 214–232. Springer, Heidelberg (2007)

23. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassambly. In: 10-th ACM Conference on Computer and Communication Security, pp. 290–299 (2003)
24. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004)
25. Ogiso, T., Sakabe, Y., Soshi, M., Miyaji, A.: Software obfuscation on a theoretical basis and its implementation. IEEE Trans. Fundamentals E86-A(1) (2003)
26. Ostrovsky, R., Skeith, W.E.: Private searching on streaming data. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 223–240. Springer, Heidelberg (2005)
27. Szor, P., Ferrie, P.: Hunting for metamorphic. In: 2001 Virus Bulletin Conference, pp. 123–144 (2001)
28. Valiant, L.: A theory of learnable. Communications of the ACM 27(11), 1134–1142 (1984)
29. Varnovsky, N.P., Zakharov, V.A.: On the possibility of provably secure obfuscating programs. In: Broy, M., Zamulin, A.V. (eds.) PSI 2003. LNCS, vol. 2890, pp. 91–102. Springer, Heidelberg (2004)
30. Varnovsky, N.P.: A note on the concept of obfuscation. In: Proceedings of Institute for System Programming, Moscow, vol. 6, pp. 127–137 (2004)
31. Wang, C., Davidson, J., Hill, J., Knight, J.: Protection of software-based survivability mechanisms. In: International Conference of Dependable Systems and Networks (2001)
32. Wee, H.: On obfuscating point functions. In: 37-th Symposium on Theory of Computing, pp. 523–532 (2005)

Specifying Imperative Data Obfuscations

Stephen Drape, Clark Thomborson, and Anirban Majumdar

Department of Computer Science,
The University of Auckland, New Zealand
{stephen, cthombor, anirban}@cs.auckland.ac.nz

Abstract. An obfuscation aims to transform a program, without affecting the functionality, so that some secret information within the program can be hidden for as long as possible from an adversary. Proving that an obfuscating transform is correct (*i.e.* it preserves functionality) is considered to be a challenging task.

In this paper we show how data refinement can be used to specify imperative data obfuscations. An advantage of this approach is that we can establish a framework in which we can prove the correctness of our obfuscations. We demonstrate our framework by considering some examples from obfuscation literature. We show how to specify these obfuscations, prove that they are correct and produce generalisations.

Keywords: Data Obfuscation, Refinement, Specification, Correctness.

1 Introduction

Skype's internet telephony client [2], SDC Java DRM (according to [11]), and most software license-control systems rely heavily on obfuscation for their security. After the landmark proof of Barak *et al.* [1], there seems little hope of designing a perfectly-secure software black-box, for any broad class of programs. To date, no one has devised an alternative to Barak's model, in which we would be able to derive proofs of security for systems of practical interest. These theoretical difficulties do not lessen practical interest in obfuscation, nor should it prevent us from placing appropriate levels of reliance on obfuscated systems in cases where the alternative of a hardware black-box is infeasible or uneconomic.

In this paper we define obfuscation as a heuristic method whose objective is to transform a program, without affecting relevant aspects of its functionality, in such a way that some secret information in the program can be preserved as long as possible from some set of adversaries. The second clause in our objective implies that theoretical study of the effectiveness of an obfuscation will be impossible until we have a validated, and theoretically-tractable, model of adversarial attack. The first clause is, by contrast, an appropriate domain for theoretical study. We expect our compilers to accurately preserve program semantics when they transform our source codes into object codes. We have a similar expectation of obfuscating compilers and object-code obfuscators. Theoretical study of the correctness of obfuscating systems is as yet in its infancy. In this paper we describe a promising approach for specifying obfuscating transforms for imperative

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 299–314, 2007.

languages. Our approach allows us to establish a framework for the proving the correctness of data obfuscations which we illustrate by constructing correctness proofs for some examples from [3] and [8] which were stated without proof.

In [7] obfuscation is considered to be data refinement [6] and obfuscations for abstract data-types were developed using a functional language (Haskell [10]). In this paper we extend the data refinement approach to imperative data obfuscations. We will consider programs consisting of assignments, conditionals and loops and we model these statements as functions that change the state. Thus we will be able to specify data obfuscations as functional refinements. As a consequence, we find that not only can we prove the correctness of *all* our data obfuscations but we can also specify more general obfuscations and study the effects of composing different obfuscating transforms. Thus our approach may someday be used as a method for generating obfuscated programs.

As stated earlier an obfuscation should preserve some secret information but what do we mean by this? In [7] a function (operation) was said to be obfuscated if it is harder to prove properties (*i.e.* assertions) about the function. Thus, in that case, the goal of obfuscation was to keep a set of assertions secret. It is beyond the scope of this paper to develop this notion for imperative programs — however we would expect that it is harder to prove assertions about a data obfuscated program if, for instance, it has more variables (that are not just dummy or temporary variables) or that the expressions that are used to compute values of variables are more complicated.

2 Creating a Specification Framework

In this section we show how to specify imperative data obfuscations as data refinements by modelling imperative statements as functions on the state.

2.1 Modelling Statements as Functions

We define a *statement* to be a function on states: $Statement :: State \rightarrow State$ where a state is defined to be a set of mappings from variables to values (or expressions computing values). We assume that the variables are integer valued and any expressions consist of arbitrary-precision arithmetic operators. We concentrate on code fragments with no methods, exceptions or pointers.

Suppose that we have a set of states $\mathcal{S}$. For some initial state $\sigma_0 \in \mathcal{S}$ and some statement T, the effect of statement T on σ_0 is to produce a new state $\sigma_1 \in \mathcal{S}$ such that $\sigma_1 = T(\sigma_0)$. Suppose that we have a sequential composition (;) of statements, which we will call a *block*, $B = T_1; T_2; \ldots; T_n$. If the initial state is σ_0 then the final state σ_n is given by

$$\sigma_n = B(\sigma_0) = T_n\left(\ldots T_2\left(T_1\left(\sigma_0\right)\right)\ldots\right)$$

For our simple language, we consider the following statement types: *skip*, *assignments* (*var* := *expr*), *conditionals* (**if** *pred* **then** *statements* **else** *statements*) and *loops* (**while** *pred* **do** *statements*).

The statement *skip* does not change the state and so if $S \equiv skip$ then $S(\sigma_0) = \sigma_0$. For an assignment A of the form $A \equiv x := e$, if the initial state contains the mapping $x \mapsto x_0$ then the state after the assignment will have a mapping $x \mapsto e$. Note that if the expression e is a function of x, say $f(x)$, then the mapping $x \mapsto x_0$ will be replaced by the mapping $x \mapsto f(x_0)$. As an alternative, if $x \mapsto x_0$ is the initial mapping for x then we can write

$$A(\sigma_0) = \sigma_0 \oplus \{x \mapsto e[x_0/x]\}$$

using functional overriding ($\oplus$) and substitution ($/$).

A conditional statement C has the form $C \equiv \textbf{if } p \textbf{ then } T \textbf{ else } E$ where p is a predicate with type $p :: State \to \mathbb{B}$ and T and E are blocks. Then for some initial state σ_0 we have that

$$C(\sigma_0) = \begin{cases} T(\sigma_0) & \text{if } p(\sigma_0) \\ E(\sigma_0) & \text{otherwise} \end{cases}$$

A loop statement L has the form $L \equiv \textbf{while } p \textbf{ do } T$ where p is a predicate and T is a block. Then for some initial state σ_0 we have that

$$L(\sigma_0) = T^i(\sigma_0) \text{ where } i = min\{i :: \mathbb{N} \mid p\,(T^i(\sigma_0)) = False\}$$

Note that this minimum does not exist if the loop fails to terminate.

2.2 Using Refinement

The obfuscations that we will consider in this paper are data obfuscations and we will suppose that such an obfuscation will act on a state σ to produce a new state $\mathcal{O}(\sigma)$. Thus we can consider the set of states $\mathcal{S}$ to be obfuscated to produce a new set of states $\mathcal{O}(\mathcal{S})$. To specify a data obfuscation $\mathcal{O}$ we will supply a function $cf :: State \to State$ which we call the *conversion function* which satisfies

$$cf(\sigma) = \sigma' \Rightarrow \sigma \in \mathcal{S} \wedge \sigma' \in \mathcal{O}(\mathcal{S})$$

Note that the type of the conversion function is the same as the type of a statement and so cf usually takes the form of an assignment.

For a (functional) refinement we require an abstraction function af with type $af :: State \to State$ which maps a state from $\mathcal{O}(\mathcal{S})$ to a state from $\mathcal{S}$ and is a post sequential inverse for cf (*i.e.* $cf; af \equiv skip$). As well as an abstraction function, for refinement, we need an invariant I on the obfuscated state such that for states $\sigma \in \mathcal{S}$ and $\mathcal{O}(\sigma) \in \mathcal{O}(\mathcal{S})$

$$\sigma \rightsquigarrow \mathcal{O}(\sigma) \Leftrightarrow (\sigma = af(\mathcal{O}(\sigma))) \wedge I(\mathcal{O}(\sigma)) \tag{1}$$

Note that for most of our transformations unless otherwise stated $I \equiv True$. The expression "$\sigma \rightsquigarrow \mathcal{O}(\sigma)$" means that the state σ is obfuscated (refined) into $\mathcal{O}(\sigma)$. Using the conversion function we have that $cf(\sigma) = \mathcal{O}(\sigma) \Rightarrow \sigma \rightsquigarrow \mathcal{O}(\sigma)$.

Suppose that we have a block B and we want to obfuscate it using data refinement to obtain a block $\mathcal{O}(B)$ which preserves the correctness of B. We say that $\mathcal{O}(B)$ is *correct* (with respect to B) under the obfuscation $\mathcal{O}$ if it satisfies

$$(\forall \sigma \in \mathcal{S}) \bullet \sigma \rightsquigarrow \mathcal{O}(\sigma) \Rightarrow B(\sigma) \rightsquigarrow \mathcal{O}(B)(\mathcal{O}(\sigma)) \tag{2}$$

Using Equation (1) we obtain the following equation:

$$af;\ B \equiv \mathcal{O}(B);\ af \tag{3}$$

By writing the abstraction function as a statement we can construct two blocks $af; B$ and $\mathcal{O}(B); af$ and proving the equivalence of these blocks establishes that $\mathcal{O}(B)$ is correct.

Since $cf; af \equiv skip$ an alternative correctness equation can be obtained by pre-composing Equation (3) by cf:

$$B \equiv cf;\ \mathcal{O}(B);\ af \tag{4}$$

If we also have that $af; cf \equiv skip$ then we can post compose Equation (3) by cf to obtain

$$\mathcal{O}(B) \equiv af;\ B;\ cf \tag{5}$$

In Appendix A we discuss in detail how to use these equations to construct proofs of correctness for imperative obfuscations.

2.3 Obfuscating Statements

Suppose that we have a data obfuscation that changes a variable x using a conversion function cf and abstraction function af satisfying $cf; af \equiv skip$. This means that af and cf are statements of the form

$$af \equiv x := G(x) \qquad cf \equiv x := F(x)$$

for some functions F and G.

Suppose we have an obfuscation for x (with cf and af defined as above) then let us consider the statement $P_1 \equiv x := e$ where e is an expression that may contain an occurrence of x. We have that:

$$\mathcal{O}(x := e) \equiv x := F(e') \text{ where } e' = e[G(x)/x] \tag{6}$$

For example, the expression $x := x+1$ would be transformed to $x := F(G(x)+1)$. Note that the expression $e[G(x)/x]$ denotes how a use of x is obfuscated.

Now let us suppose that $P_2 \equiv$ **if** $p(x)$ **then** T **else** E for some predicate p (which depends on a variable x) and blocks T and E. We propose that

$$\mathcal{O}(P_2) \equiv \textbf{if } p[G(x)/x] \textbf{ then } \{af; T; cf\} \textbf{ else } \{af; E; cf\}$$

with af as above. Using Equation (3) we can show that $\mathcal{O}(P_2);\ af \equiv af;\ P_2$.

Thus, since $cf; af \equiv skip$ (and using the definition of $\mathcal{O}$) then

$$\mathcal{O}(\textbf{if } p \textbf{ then } T \textbf{ else } E) \equiv \textbf{if } \mathcal{O}(p) \textbf{ then } \mathcal{O}(T) \textbf{ else } \mathcal{O}(E) \tag{7}$$

Finally, suppose that $P_3 \equiv \textbf{while } p(x) \textbf{ do } S$ then we propose that

$$\mathcal{O}(P_3) \equiv \textbf{while } p[G(x)/x] \textbf{ do } \{af; S; cf\}$$

with af as above. For the correctness of $\mathcal{O}(P_3)$ we need to show that $af; P_3 \equiv \mathcal{O}(P_3); af$ and so for the LHS of the identity we will need to "move" af. In executing $af; P_3$ we will obtain a block of the form $af; S^n$ where $n :: \mathbb{N}$. Since $cf; af \equiv skip$ then

$$af; S^n \equiv (af; S; cf)^n; af$$

To move af through the **while** loop we need to change the expression for the guard. When af is before the loop we have an assignment to x and so this assignment needs to put into the guard and so the guard becomes $p[G(x)/x]$. Now $af; S; cf$ is a refinement (obfuscation) of S with respect to af and so the value of x is obfuscated while the loop is executed — thus the change to the guard is correct as the predicate p will need the original value of x. Thus, since $cf; af \equiv skip$, we have shown that

$$af; \textbf{while } p(x) \textbf{ do } S \equiv \textbf{while } p[G(x)/x] \textbf{ do } \{af; S; cf\}; af \tag{8}$$

Suppose that we want to obfuscate a sequential composition of blocks. Let B_1 and B_2 be two blocks of code and by using Equation (3) we can show that

$$\mathcal{O}(B_1;\ B_2) \equiv \mathcal{O}(B_1);\ \mathcal{O}(B_2) \tag{9}$$

So when applying a data obfuscation to a sequence of statements (blocks) we can obfuscate each statement (block) individually and compose the results.

3 Variable Transformations

We now give some examples of data transformations that can be used to obfuscate variables.

3.1 Encoding

In [3] an obfuscation for variables called an *encoding* is given. A simple example of an encoding for some variable x is $x \rightsquigarrow \alpha * x + \beta$ where α and β are constants. For this transformation we have the following refinement functions:

$$cf \equiv x := \alpha * x + \beta \qquad af \equiv x := (x - \beta)/\alpha$$

For exact arithmetic, we have that $cf; af \equiv skip \equiv af; cf$. The conversion and abstraction functions are of the form of the functions used in Section 2.3 and so we can use the equations given in that section for transforming statements.

In [8], the following example was discussed:

$$P \equiv \{i := 1;\ s := 0;\ \textbf{while}\ (i < 15)\ \textbf{do}\ \{s := s + i;\ i := i + 1\}\}$$

This example was then converted using the mapping $i \rightsquigarrow 2 * i$ to give:

$$\mathcal{O}(P) \equiv \{i := 2;\ s := 0;\ \textbf{while}\ (i < 30)\ \textbf{do}\ \{s := s + (i/2);\ i := i + 2\}\}$$

This transformation was given without a proof of correctness. The refinement functions for this obfuscation are:

$$cf \equiv i := 2 * i \qquad af \equiv i := i/2$$

To prove that $\mathcal{O}(P)$ is correct we use Equation (3) to show that: $af; P \equiv \mathcal{O}(P); af$. This proof is given in Appendix A.3.

A more complicated variable transformation for x can be obtained by using $cf \equiv x := \alpha * x + \beta * y$ where α and β are constants and y is a program variable.

3.2 Variable Splitting

Another variable transformation mentioned in [3] is the concept of variable splitting. This is where a variable x (say) is represented by two or more new variables so that the information contained in x is "split" across these new variables. For an example transformation, we will split the integer variable x into two new integers variables a and b such that $a = x$ div 10 and $b = x$ mod 10. We can write the conversion and abstraction functions as follows:

$$af \equiv x := 10 * a + b \qquad cf \equiv \{a := x \text{ div } 10;\ \ b := x \text{ mod } 10\}$$

For this transformation we have the invariant $I \equiv 0 \leq b \leq 9$. This invariant ensures that the definition of af is valid and if this invariant holds then $cf; af \equiv skip$ and $af; cf \equiv skip$.

Under this transformation, the assignment $x{+}{+}$ (*i.e.* $x := x + 1$) becomes:

$$a := (10 * a + b + 1) \text{ div } 10;\ \ b := (b + 1) \text{ mod } 10$$

Note that $((10 * a) + b + 1) \text{ mod } 10 \equiv (b + 1) \text{ mod } 10$.

As an alternative, we propose that a correct transformation of $S \equiv x{+}{+}$ is

$$\mathcal{O}(S) \equiv \textbf{if}\ (b == 9)\ \textbf{then}\ \{a := a + 1;\ b := 0\}\ \textbf{else}\ \{b := b + 1\}$$

and this can be proved correct by showing that $S \equiv cf; \mathcal{O}(S); af$. The advantage of the latter transformation is that it does not have traces of the abstraction and conversion functions. The proof is given in Appendix A.4.

4 Array Transformations

Various array transformations are mentioned in [3] such as: *Folding* (1-D arrays are transformed into n-D arrays), *Flattening* (n-D arrays are changed into 1-D arrays), *Splitting* (one array is transformed into two or more arrays) and *Merging* (two or more arrays are combined into one array).

4.1 Changing Array Indices

Folding and flattening can be considered to be transformations that change an array index — how can we specify these transformations? For example, suppose that we have an array A of size n and an array R of size $p \times q$ (where $p \times q = n$). One way to convert between A and R is to use the transformation $A[i] := R[i \text{ div } q, i \bmod q]$ which has the inverse $R[j,k] := A[j * q + k]$.

How can we write a conversion function for this kind of transformation? We need to consider how $A[0], A[1], \ldots, A[n-1]$ are all transformed. So we could give a set of n transformations (one for each element of the array) but in a program the index for an array is usually a variable (or an expression). Thus we need to write an expression for $A[j]$, where $j \in [0..n)$, which shows how the array is transformed. Note that this is not a variable transformation of j as j is merely a dummy variable acting as a placeholder. When using such an expression at a particular point we need to instantiate j with the expression for the array index.

If we want to transform the array A into the array R using an index change function f then the conversion and abstraction functions are:

$$cf \equiv R[f(j)] := A[j] \qquad \text{and} \qquad af \equiv A[j] := R[f(j)]$$

Suppose that we want to transform the statement $A[i] := A[i-1] + 1$. From Equation (5) we have:

$$A[j] := R[f(j)]; \quad A[i] := A[i-1] + 1; \quad R[f(j)] := A[j]$$

Using the proof techniques discussed in Appendix A we can reduce the set of statements to:

$$R[f(i)] := R[f(i-1)] + 1$$

4.2 Array Splitting

An array split aims to split an array A (of size n) into two new arrays P (of size m_p) and Q (of size m_q). This idea was generalised in [8] as follows. For an array split which uses two new arrays, we need three functions c (called the *choice function*), f_p and f_q (these functions determine the positions of the elements in each of the arrays). The types of the functions are as follows:

$$c :: [0..n) \to \mathbb{B} \qquad f_p :: [0..n) \to [0..m_p) \qquad f_q :: [0..n) \to [0..m_q)$$

The relationship between A and P and Q is given by the following rule:

$$A[i] = \begin{cases} P[f_p(i)] & \text{if } c(i) \\ Q[f_q(i)] & \text{otherwise} \end{cases}$$

Note that we can only apply this transformation to statements that use A with an index. For example, we could not easily transform statements which pass the array A to other methods.

In [3], an example array split was given in which one of the new arrays contained the elements of A, in order, which had an even index and the other array contained the rest of the elements. For this split, we can define

$$c = (\lambda i.i \,\%\, 2 == 0) \qquad f_p = f_q = (\lambda i.i/2)$$

In [7], a program for producing Fibonacci numbers using arrays was obfuscated using the example array split from [3]. For this obfuscation, the statement

$$S \equiv A[i] := A[i-1] + A[i-2] \tag{10}$$

was transformed to:

$$\begin{array}{r} \textbf{if } (i \,\%\, 2 == 0) \textbf{ then } P[i/2] := Q[(i-1)/2] + P[(i-2)/2] \\ \textbf{else } Q[i/2] := P[(i-1)/2] + Q[(i-2)/2] \end{array} \tag{11}$$

Is this transformation correct? Let us show how to derive a correct obfuscation for the statement (10) using the generalised array split.

We can write a conversion function for the generalised array split as follows

$$cf \equiv \textbf{if } (c(j)) \textbf{ then } P[f_p(j)] := A[j] \textbf{ else } Q[f_q(j)] := A[j]$$

and so the abstraction function can be written as

$$af \equiv \textbf{if } (c(j)) \textbf{ then } A[j] := P[f_p(j)] \textbf{ else } A[j] := Q[f_q(j)]$$

We can show that $cf; af \equiv skip \equiv af; cf$. Note that when we use these functions we will have to instantiate the index j to a particular value (or expression). To derive a correct obfuscation for S (in Equation (10)) we can use Equation (5) to compute $af; S; cf$. A sketch of the derivation can be seen in Appendix A.5 which gives the general form for $\mathcal{O}(S)$ as:

$$\begin{array}{l} \textbf{if } (c(i)) \textbf{ then } \{\textbf{if } (c(i-1)) \\ \quad \textbf{then } \{\textbf{if } (c(i-2)) \textbf{ then } P[f_p(i)] := P[f_p(i-1)] + P[f_p(i-2)] \\ \qquad\qquad\qquad \textbf{else } P[f_p(i)] := P[f_p(i-1)] + Q[f_q(i-2)]\} \\ \quad \textbf{else } \{\textbf{if } (c(i-2)) \textbf{ then } P[f_p(i)] := Q[f_q(i-1)] + P[f_p(i-2)] \\ \qquad\qquad\qquad \textbf{else } P[f_p(i)] := Q[f_q(i-1)] + Q[f_q(i-2)]\}\} \\ \textbf{else } \{\textbf{if } (c(i-1)) \textbf{ then } \{\textbf{if } (c(i-2)) \\ \qquad\qquad\qquad\qquad \textbf{then } Q[f_q(i)] := P[f_p(i-1)] + P[f_p(i-2)] \\ \qquad\qquad\qquad\qquad \textbf{else } Q[f_q(i)] := P[f_p(i-1)] + Q[f_q(i-2)]\} \\ \qquad\qquad\qquad \textbf{else } \{\textbf{if } (c(i-2)) \\ \qquad\qquad\qquad\qquad \textbf{then } Q[f_q(i)] := Q[f_q(i-1)] + P[f_p(i-2)] \\ \qquad\qquad\qquad\qquad \textbf{else } Q[f_q(i)] := Q[f_q(i-1)] + Q[f_q(i-2)]\}\} \end{array}$$

We can simplify the expression for $\mathcal{O}(S)$ for this split by removing infeasible paths. For example, when we have a statement of the form:

$$\textbf{if } (c(i)) \textbf{ then } \{\textbf{if } (c(i-1)) \textbf{ then } X \textbf{ else } Y\}$$

then X cannot be reached since $c = (\lambda i.i\,\%\,2 == 0)$ and when $c(i)$ is True then $c(i-1)$ must be False. By removing all the infeasible paths and substituting the functions c, f_p and f_q we obtain

$$\begin{array}{r} \textbf{if } (i\,\%\,2 == 0) \textbf{ then } P[i/2] := Q[(i-1)/2] + P[(i-2)/2] \\ \textbf{else } Q[i/2] := P[(i-1)/2] + Q[(i-2)/2] \end{array}$$

Thus the transformation given in [7] was correct.

5 Applying Data Obfuscations

In the previous sections we have given examples of data obfuscations and in this section we demonstrate some of the choices that we can make when applying our data obfuscations.

5.1 Program Blocks

If we have a piece of code $P \equiv B_1; B_2$ (where B_1 and B_2 are blocks) and an obfuscation $\mathcal{O}$ with conversion function cf and abstraction function af that satisfy $cf; af \equiv skip \equiv af; cf$. Using Equations (5) and (9) we have two ways to obfuscate P. Either we can obfuscate B_1 and B_2 separately and compose the results or we can obfuscate both blocks together *i.e.*

$$\mathcal{O}(P) \equiv \{af; B_1; cf\}; \{af; B_2; cf\} \quad \text{or} \quad \mathcal{O}(P) \equiv af; \{B_1; B_2\}; cf$$

The two obfuscations that we obtain are equivalent but they may look different. In particular the second derivation may reduce the number of assignments.

For example, suppose that $P \equiv \{x := x+1;\ B;\ x := 3*x\}$ where B is a block of code in which x does not occur and $cf \equiv x := x + 2$ and $af \equiv x := x - 2$. If we obfuscate the two assignments separately then we have that

$$\mathcal{O}(P) \equiv \{x := x + 1;\ B;\ x := 3 * x - 4\}$$

However computing $af;\ P;\ cf$ will give us the following set of simultaneous equations (see Appendix A for more details how to compute this set):

$$x_1 = x_0 - 2;\ \ x_2 = x_1 + 1;\ \ B;\ \ x_3 = 3 * x_2;\ \ x_4 = x_3 + 2 \tag{12}$$

Reducing this set of equations (with x not occurring in B) gives us:

$$B;\ \ x_4 = 3 * (x_0 - 1) + 2$$

Thus $\mathcal{O}(P) \equiv \{B;\ x := 3 * x - 1\}$.

The two derivations produce equivalent programs but the second program only has one assignment to x. From an obfuscation point of view, the first program would appear to be better as it has more assignments to x and so it is (slightly) harder to work out the value of x at the end of $\mathcal{O}(P)$.

Instead of completely reducing a set of simultaneous equations we can partially reduce them. For instance for the set of equations in (12), we can substitute x_1 and x_3 to obtain:

$$\mathcal{O}(P) \equiv \{x := x - 1;\ B;\ x := 3 * x + 2\}$$

Thus we have some flexibility when deriving obfuscation for a sequence of statements using a particular conversion function.

5.2 Combining Transformations

Since we are considering our obfuscations as functions we may naturally want to compose obfuscations. For some variable x suppose that we have two obfuscations $\mathcal{O}_1$ and $\mathcal{O}_2$ with conversion functions $cf_1 \equiv x := f_1(x)$ and $cf_2 \equiv x := f_2(x)$ and corresponding abstraction functions $af_1 \equiv x := g_1(x)$ and $af_2 \equiv x := g_2(x)$. To obfuscate a statement S by applying $\mathcal{O}_1$ followed by $\mathcal{O}_2$ we have:

$$\mathcal{O}_2(\mathcal{O}_1(S)) \equiv af_2;\ af_1;\ S;\ cf_1;\ cf_2$$

This is equivalent to having a single obfuscation $\mathcal{O}_{1;2}$ with conversion function $cf_{1;2} \equiv x := (f_2 \cdot f_1)(x)$ and abstraction function $af_{1;2} \equiv x := (g_1 \cdot g_2)(x)$. We can define $\mathcal{O}_{1;2} \equiv \mathcal{O}_2 \cdot \mathcal{O}_1$.

For example, we can combine a variable transformation with an array obfuscation given in Section 4.1. For instance if we had the functions $f :: \mathbb{Z} \to \mathbb{Z}$ and $p :: [0..n) \to [0..n)$ (with appropriate inverses) then a possible conversion function is $cf \equiv A[i] := f(A[p(i)])$ in which f acts as a variable transformation and p is an array index permutation.

6 Conclusion

In this paper we have extended work from [7] and considered imperative data obfuscations as data refinements. By using functional refinement and modelling statements as functions on the state we were able to prove the correctness of imperative data obfuscations, including some of the data obfuscations from [3,8] which were stated without proof. For data refinement we give functions (the conversion and abstraction functions) describing the relationship between the before and after states of a obfuscated variable. Using these functions we can prove that a sequence of statements has been correctly obfuscated. Initially we considered simple variable obfuscations and then we showed how to extend our work to deal with more complicated obfuscations such as array transformations. In Section 5 we saw that we often have a choice about how we can apply data obfuscations such as using single statements vs. blocks of statements or reducing a set of simultaneous equations differently. Thus, by applying a data obfuscation to the same piece of code in different ways, we can produce different obfuscations.

Our purpose in this contribution has not been to propose new obfuscating transforms but to show a way to specify and prove existing transforms. This, we

believe, is an important step towards ensuring that the obfuscated program and its unobfuscated counterpart are functionally equivalent (same I/O behaviour) after the obfuscating transforms are applied. The simple program constructs that we have targetted form the basis for all imperative languages and therefore our method is generic enough to be applicable to a wider class of imperative languages (we chose not to target language-specific constructs). An application of using our framework for specifying and proving correctness of obfuscations can been seen in design of *slicing obfuscations* [9], which are used to impede static program analysis with a slicer (which can be used as a tool by an adversary to reverse engineer programs [3]).

One drawback with our method for producing data obfuscations is that the conversion and abstraction function can remain visible in the code. To prevent this we can try to combine these functions with surrounding statements. Sometimes our obfuscations may need extra assignments, temporary variables and extra computations. Thus we may have a trade-off between the efficiency and the complexity of our obfuscations. We should ensure that our obfuscations do not adversely affect the efficiency of our programs and so we may need to restrict how complicated we make our obfuscations. Ways to do this has been shown in [9]. Also, it would seem that an optimizing compiler will effectively strip out the conversion and abstraction functions in the code if they are trivially analysable by static analysis. We argue that this is not an immediate concern to us since commercially distributable software will be obfuscated after the optimization phase of the developer's compiler.

We made various restrictions and an area for future work would be to see how these restrictions could be removed. We used only arbitrary-precision arithmetic but if we relaxed this restriction we may not be able to use obfuscations such as $x \rightsquigarrow \alpha * x + \beta$ since this may cause x to overflow and we may not be able to construct an inverse (as it requires division). All the obfuscations that we have considered have been data obfuscations but another class of obfuscations is *control flow obfuscations* (for example, using predicates [4] and control flow flattening [12]). Can control flow obfuscations be specified using refinement?

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, pp. 1–18. Springer, Heidelberg (2001)
2. Biondi, P., Desclaux, F.: Silver needle in the skype. Presentation at BlackHat Europe, March, Slides available from (2006), URL http://www.blackhat.com/html/bh-media-archives/bh-archives-2006.html
3. Collberg, C., Thomborson, C., Low, C.: A taxonomy of obfuscating transformations. Technical Report 148, Department of Computer Science, University of Auckland (July 1997)
4. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient and stealthy opaque constructs. In: ACM SIGACT Symposium on Principles of Programming Languages, pp. 184–196. ACM Press, New York (1998)

5. Cytron, R., Ferrante, J., Rosen, B.K., Wegman, M.N., Zadeck, F.K.: Efficiently computing static single assignment form and the control dependence graph. ACM Transactions on Programming Languages and Systems 13(4), 451–490 (1991)
6. de Roever, W.-P., Engelhardt, K.: Data Refinement: Model-Oriented Proof Methods and their Comparison. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (1998)
7. Drape, S.: Obfuscation of Abstract Data-Types. DPhil thesis, Oxford University Computing Laboratory (2004)
8. Drape, S., de Moor, O., Sittampalam, G.: Transforming the .NET Intermediate Language using Path Logic Programming. In: Principles and Practice of Declarative Programming, pp. 133–144. ACM Press, New York (2002)
9. Drape, S., Majumdar, A.: Design and Evaluation of Slicing Obfuscations. Technical Report 311, Department of Computer Science, The University of Auckland, New Zealand (June 2007)
10. Peyton Jones, S.: The Haskell 98 language and libraries: the revised report. Journal of Functional Programming 13(1) (2003)
11. Santos, N., Pereira, P., eSilva, L.M.: A Generic DRM Framework for J2ME Applications. In: Pitkänen, O. (ed.) First International Mobile IPR Workshop: Rights Management of Information (MobileIPR), August 2003. Helsinki Institute for Information Tecnhology, pp. 53–66 (2003)
12. Wang, C., Hill, J., Knight, J.C., Davidson, J.W.: Protection of software-based survivability mechanisms. In: DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS), pp. 193–202. IEEE Computer Society Press, Los Alamitos (2001)

A Proofs of Correctness

In the main part of the paper we gave a way of specifying imperative data obfuscations. We now discuss how to use this specification to construct correctness proofs for some of our example obfuscations.

A.1 Simultaneous Equations

Suppose that we obfuscate S to obtain $\mathcal{O}(S)$ where af and cf are the abstraction and conversion functions for the obfuscation. We can use Equation (4) to prove that $\mathcal{O}(S)$ is a correct obfuscation of S by showing that the sequence of statements cf; $\mathcal{O}(S)$; af is equivalent to S. Suppose that we have an obfuscation that transforms a variable x (say) then this proof could take the form:

$$x := f(x); \quad x := u(x); \quad x := g(x)$$

where f, g and u are functions. To simplify this expression we can substitute values of x in sequential order by rewriting the sequence of statements as a set of simultaneous equations. Each definition of a variable will have a different name which is usually the name of the variable with a subscript (*e.g.* x_2) and we will use the convention that the initial value of a variable has a subscript 0. All the uses of a variable are renamed to correspond to the appropriate assignment.

The sequence above can be rewritten as the following set of equations:

$$x_1 = f(x_0); \quad x_2 = u(x_1); \quad x_3 = g(x_2)$$

To distinguish between programs and sets of equations, whenever we convert to a set of simultaneous equation we will use the convention that the assignment symbol := is replaced by equality =. By substituting the values for x_1 and x_2 we obtain the following:

$$x_1 = f(x_0); \quad x_2 = u(f(x_0)); \quad x_3 = g(u(f(x_0)))$$

We can remove the assignments for x_1 and x_2 as they are now redundant. So now we have $x_3 = g(u(f(x_0))$ which corresponds to the statement $x := g(u(f(x)))$.

This conversion from assignments to simultaneous equations is similar to converting code to SSA (Static Single Assignment) form which is often used in conjunction with compiler optimisations (for example, [5] gives details about how to compute SSA form). In SSA form, each definition of a variable is given a different name and each use is renamed according to the appropriate definition. When there are different control flow paths, a special statement called a ϕ (phi) function is added. However, as we are only aiming to simplify a set of simultaneous equations, we will not use the SSA form directly. In particular, our proofs will not need to use phi functions as we will use the results of Section 2.3 to enable us to deal with **if** and **while** separately and we can obfuscate a sequence of statements by obfuscating the individual statements. We will only use the SSA form as a guide to help us to specify a set of simultaneous equations which we can manipulate and simplify.

A.2 Steps in a Proof

There are four main steps in constructing our proofs of correctness.

Simultaneous Equations. The first step is to convert a sequential program into a set of simultaneous equations using the SSA form as a guide. This means that each new definition of a variable has a unique subscript and each use of a variable should refer to the last instance of the variable.

Substitution. Once we have converted our sequential code to a set of simultaneous equations then the next phase is to reduce the set of equations by performing substitutions and in particular we want to hide the occurrence of the conversion and abstraction functions. However, sometimes problems can arise.

Suppose that we have the following set of simultaneous equations:

$$y_1 = x_0 + 1; \quad x_1 = x_0 + 2; \quad y_2 = y_1 - 1$$

Substituting the value for y_1 gives

$$y_1 = x_0 + 1; \quad x_1 = x_0 + 2; \quad y_2 = x_0$$

We can see that the "last" definition for x is at x_1 but the expression for y_2 uses an earlier definition of x. Whenever this type of situation occurs then we cannot immediately convert such sets of equations back to sequential code. The last step discusses possible solutions for this problem.

Redundant Definitions. We would like to remove traces of the conversion and abstraction functions and so the next step after substitution is to remove redundant definitions. A definition $x_i := e$ is *redundant* in a set of simultaneous equations if no equation uses x_i and there exists some definition $x_j := e'$ where $j > i$. This last condition ensures that we do not remove the "last" definition of a variable (and since we convert using a form of SSA we know that the last definition of a variable will have the largest subscript).

Converting back. Once the set of simultaneous equations has been reduced they need to be converted to sequential code. As mentioned earlier, sometimes we cannot immediately convert the set of equations back to sequential code. For example, suppose that after substitution and refinement we are left with the following pair of simultaneous equations:

$$x_1 = x_0 + 2; \quad y_2 = x_0$$

This cannot be converted to:

$$x := x + 2; \quad y := x$$

as the final value of y in this sequence is equivalent to x_1 not x_0 as required. One solution is to introduce a new variable which holds the value of x_0:

$$t_1 = x_0; \quad x_1 = x_0 + 2; \quad y_2 = t_1$$

So this can be converted to:

$$t := x; \quad x := x + 2; \quad y := t$$

A.3 A Loop Proof

In Section 3.1 a variable encoding was used in a **while** loop. Here is the proof of correctness to show that $af;\ P \equiv \mathcal{O}(P);\ af$.

$$
\begin{array}{ll}
 & af; P \\
\equiv & \{\text{definitions}\} \\
 & af;\ \ i := 1;\ \ s := 0;\ \textbf{while}\ (i < 15)\ \textbf{do}\ \{s := s + i;\ i := i + 1\} \\
\equiv & \{cf; af \equiv skip\} \\
 & af;\ i := 1;\ s := 0;\ cf;\ af;\ \textbf{while}\ (i < 15)\ \textbf{do}\ \{s := s + i;\ i := i + 1\} \\
\equiv & \{\text{Equations (5) and (6)}\} \\
 & i := 2;\ \ s := 0;\ \ af;\ \textbf{while}\ (i < 15)\ \textbf{do}\ \{s := s + i;\ i := i + 1\} \\
\equiv & \{\text{Equation (8)}\} \\
 & i := 2;\ s := 0;\ \textbf{while}\ ((i/2) < 15)\ \textbf{do}\{af;\ s := s + i;\ i := i + 1;\ cf\};\ af \\
\equiv & \{\text{Equation (6)}\} \\
 & i := 2;\ s := 0;\ \textbf{while}\ ((i/2) < 15)\ \textbf{do}\{s := s + (i/2);\ i := 2 * ((i/2) + 1)\};\ af
\end{array}
$$

$\equiv$ {exact arithmetic}
$i := 2;\ s := 0;\ \textbf{while}\ (i < 30)\ \textbf{do}\ \{s := s + i/2;\ i := i + 2\};\ af$
$\equiv$ {definitions}
$\mathcal{O}(P);\ af$

A.4 Variable Split

In Section 3.2 we give a transformation for the statement $S \equiv x{+}{+}$. We show that $cf;\ \mathcal{O}(S);\ af \equiv S$ and so the transformation is correct.

$cf; \mathcal{O}(S); af$
$\equiv$ $\{af; cf \equiv skip\}$
$cf;\ \textbf{if}\ (b == 9)\ \textbf{then}\ \{af;\ cf;\ a := a + 1;\ b := 0;\ af;\ cf\}$
$\qquad\qquad\qquad\qquad\ \textbf{else}\ \{af;\ cf;\ b := b + 1;\ af;\ cf\};\ af$
$\equiv$ {Equation (7)}
$cf;\ af;\ \textbf{if}\ ((x \bmod 10) == 9)\ \textbf{then}\ \{cf;\ a := a + 1;\ b := 0;\ af\}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \textbf{else}\ \{cf;\ b := b + 1;\ af\};\ cf;\ af$
$\equiv$ {definitions and $cf; af \equiv skip$}
$\textbf{if}\ ((x \bmod 10) == 9)$
$\textbf{then}\ \{a := x \operatorname{div} 10;\ b := x \bmod 10;\ a := a + 1;\ b := 0;\ x := 10 * a + b\}$
$\textbf{else}\ \{a := x \operatorname{div} 10;\ b := x \bmod 10;\ b := b + 1;\ x := 10 * a + b\}$
$\equiv$ {simultaneous equations in branches}
$\textbf{if}\ ((x_0 \bmod 10) == 9)\ \textbf{then}\ \{a_1 = x_0 \operatorname{div} 10;\ b_1 = x_0 \bmod 10;\ a_2 = a_1 + 1;$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ b_2 = 0;\ x_1 = 10 * a_2 + b_2\}$
$\textbf{else}\ \{a_3 = x_0 \operatorname{div} 10;\ b_3 = x_0 \bmod 10;\ b_4 = b_3 + 1;\ x_2 = 10 * a_3 + b_4\}$
$\equiv$ {substitutions}
$\textbf{if}\ ((x_0 \bmod 10) == 9)\ \textbf{then}\ \{x_1 = 10 * (x_0 \operatorname{div} 10) + 10\}$
$\qquad\qquad\qquad\qquad\qquad\qquad\ \textbf{else}\ \{x_2 = 10 * (x_0 \operatorname{div} 10) + (x_0 \bmod 10) + 1\}$
$\equiv$ {modular arithmetic}
$\textbf{if}\ ((x_0 \bmod 10) == 9)\ \textbf{then}\ \{x_1 := x_0 + 1\}\ \textbf{else}\ \{x_2 := x_0 + 1\}$
$\equiv$ {convert back to assignments}
$\textbf{if}\ ((x \bmod 10) == 9)\ \textbf{then}\ \{x := x + 1\}\ \textbf{else}\ \{x := x + 1\}$
$\equiv$ {identical branches}
$x := x + 1$

A.5 Array Splitting

We sketch a derivation for the array transformation from Section 4.2 by computing $af; S; cf$ which, by Equation (5), is equivalent to $\mathcal{O}(S)$. Note that for arrays, when converting to a set of simultaneous equations, we use the normal

subscripts to denote new assignments on the arrays and the index i but not on the dummy variable j.

$af; S; cf$

$\equiv$ {definitions and convert to simultaneous equations}

$\textbf{if } (c(j)) \textbf{ then } A_1[j] = P_0[f_p(j)] \textbf{ else } A_1[j] = Q_0[f_q(j)];$
$A_2[i] = A_1[i-1] + A_1[i-2];$
$\textbf{if } (c(j)) \textbf{ then } P_1[f_p(j)] = A_2[j] \textbf{ else } Q_1[f_q(j)] = A_2[j]$

$\equiv$ {substitute value for A_1 with $j = i-1$ and then with $j = i-2$}

$\textbf{if } (c(j)) \textbf{ then } A_1[j] = P_0[f_p(j)] \textbf{ else } A_1[j] = Q_0[f_q(j)];$
$\textbf{if } (c(i-1)) \textbf{ then } \{\textbf{if } (c(i-2)) \textbf{ then } A_2[i] = P_0[f_p(i-1)] + P_0[f_p(i-2)]$
$\textbf{else } A_2[i] = P_0[f_p(i-1)] + Q_0[f_q(i-2)]\}$
$\textbf{else } \{\textbf{if } (c(i-2)) \ \ldots\}$
$\textbf{if } (c(j)) \textbf{ then } P_1[f_p(j)] = A_2[j] \textbf{ else } Q_1[f_q(j)] = A_2[j]$

$\equiv$ {substitute values in P_1 and Q_1 with $i = j$}

$\ldots \ \textbf{if } (c(i)) \textbf{ then } \{\textbf{if } (c(i-1)) \textbf{ then }$
$\{\textbf{if } (c(i-2)) \textbf{ then } P_1[f_p(i)] = P_0[f_p(i-1)] + P_0[f_p(i-2)] \textbf{ else } \ldots\} \textbf{ else } \{\ldots\}$
$\textbf{else } \{\textbf{if } (c(i-1)) \textbf{ then }$
$\{\textbf{if } (c(i-2)) \textbf{ then } Q_1[f_q(i)] = P_0[f_p(i-1)] + P_0[f_p(i-2)] \textbf{ else } \ldots\}$
$\textbf{else}\{\ldots\}$

$\equiv$ {sequential code (removing redundant assignments)}

$\textbf{if } (c(i)) \textbf{ then } \{\textbf{if } (c(i-1))$
$\textbf{then } \{\textbf{if } (c(i-2)) \textbf{ then } P[f_p(i)] := P[f_p(i-1)] + P[f_p(i-2)]$
$\textbf{else } P[f_p(i)] := P[f_p(i-1)] + Q[f_q(i-2)]\}$
$\textbf{else } \{\textbf{if } (c(i-2)) \textbf{ then } P[f_p(i)] := Q[f_q(i-1)] + P[f_p(i-2)]$
$\textbf{else } P[f_p(i)] := Q[f_q(i-1)] + Q[f_q(i-2)]\}\}$
$\textbf{else } \{\textbf{if } (c(i-1)) \textbf{ then } \{\textbf{if } (c(i-2))$
$\textbf{then } Q[f_q(i)] := P[f_p(i-1)] + P[f_p(i-2)]$
$\textbf{else } Q[f_q(i)] := P[f_p(i-1)] + Q[f_q(i-2)]\}$
$\textbf{else } \{\textbf{if } (c(i-2))$
$\textbf{then } Q[f_q(i)] := Q[f_q(i-1)] + P[f_p(i-2)]$
$\textbf{else } Q[f_q(i)] := Q[f_q(i-1)] + Q[f_q(i-2)]\}\}$

$\equiv$ {Equation (5)}

$\mathcal{O}(S)$

This last expression can often be simplified by removing infeasible paths.

Token-Controlled Public Key Encryption in the Standard Model

Sherman S.M. Chow

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu

Abstract. In many financial or legal scenarios (such as trading stocks, wills and safe-deposit boxes), we want to ensure that a certain task (reading the buy/sell instruction, obtaining the property, or opening the box in emergencies respectively) cannot be performed until a certain time or a certain pre-defined condition occurs. Token-controlled public key encryption (TCE), introduced in [2], is a handy tool for these situations. Roughly speaking, messages are encrypted by a public key together with a secret token in TCE, such that the receiver holding the corresponding private key cannot decrypt until the token is released. TCE is also useful in rapid distribution of information and sealed-bid auctions, etc.

In Financial Cryptography 2006, Galindo and Herranz [15] proposed a generic construction of TCE in the random oracle model. However, we show that it is insecure against insider attack, namely, a malicious user without the token can learn partial information about the message.

We propose a strengthened definition of security, and also new privacy requirements. It turns out that [15] is also insecure against outsider attack in our new definition. We then give a new generic construction provably secure in the standard model, which is nearly as efficient as a standard public key encryption scheme.

Keywords: Token-controlled, public key encryption, provable security, timed-release encryption, standard model.

1 Introduction

Token-controlled public key encryption (TCE), introduced by Baek, Safavi-Naini and Susilo [2], is an encryption scheme where the messages are encrypted by a public key together with a secret token, such that the receiver holding the corresponding private key cannot decrypt without the token. The sender has to delegate the token to a security-mediator (SEM) beforehand, whom is responsible for releasing the token when some pre-defined condition occurs.

1.1 Applications

Some motivating applications of TCE are discussed in [2] and [15]. The first scenario considers a millionaire who makes his legal will, but wants to keep

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 315–332, 2007.

it secret until the last moment. He can encrypt the wills with TCE, providing the corresponding ciphertexts to his family members, and the token to a lawyer. After the millionaire passes away, this lawyer will release the token and each of his family members can decrypt the ciphertexts and recover the wills. Another one is about the access of properties in emergencies, says deposit box of a bank. The owner can encrypt the secret which opens the box to the spouse using TCE, and instruct the bank to provide the spouse with the token in case of illness or other problems prohibiting the owner's own opening of the box. More applications related to timed-release encryption (TRE) will be described afterward.

1.2 Related Work

In [2], two constructions in the random oracle model are proposed. Their security model assumes the SEM will release the token honestly, yet can be maliciously try to break the confidentiality of the ciphertext. Galindo and Herranz [15] noted that such a security model is inadequate. A single ciphertext could be decrypted to different messages under different tokens, which means the SEM can release a "fake" token such that the decryption gives an entirely different message. To prohibit such an attack, they equipped TCE with strong existential token unforgeability. A generic construction secure in this new sense in the random oracle model is proposed. Since no real function can implement a true random oracle, a scheme with security proven in the standard model is more desirable.

1.3 Our Contributions

1. *Applications*: We give a detailed picture of the similarity between token-controlled public key encryption and timed-release encryption, clarify their differences and point out the scenarios in which the former is applicable.
2. *Stronger Definitions*: We generalize the notion of token-controlled public key encryption, in which an auxiliary message that is not controlled by the token can be included, and the token used for encryption can be different from the token delegated to the security-mediator. We also consider stronger security notions against both outsider and insider attack, and new privacy concerns including release-condition confidentiality and ciphertext unlinkability.
3. *Attacks*: We refute the security proof in [15] by showing an insider attack, namely, a malicious user can learn partial information about the message, by using only the corresponding private key, without the token. We also discuss why their scheme is also insecure against outsider attack in our new model.
4. *Constructions without Random Oracles*: We propose a new generic construction in the standard model, which is nearly as efficient as a standard public key encryption scheme. Same as previous schemes, a single token can control the decryption of more than one ciphertexts for multiple receivers, and the size of the token is small in comparison with the ciphertext size.

We give a discussion of the differences between TCE and TRE; and further applications of TCE in the next section. Section 3 briefly explains the notation

and the building blocks that will be used in the rest of this paper. In Section 4, a formal definition of TCE and its security properties are presented. We review the construction in [15] and show that their scheme is actually insecure in Section 5. Section 6 gives our new construction, together with its security and efficiency analysis. Concluding remarks are given at the end of this paper.

2 Timed-Release Encryption

An idea closely related to TCE is timed-release encryption (TRE)[1]. In TRE, a message sender encrypts a message "into the future". A trusted agent releases trapdoors at specified times which is essential for the decryption. Recent trends require TRE to be non-interactive, i.e. no communication is needed between the agent and other entities. Many applications of TRE are discussed [5,6,10,17,18]. For examples, [10] talks about asking a broker to sell a stock at a particular future time, and [17] gives an application in certified email system.

A more general notion is event-release encryption, where a sender who encrypts a message with the wish that the recipient can only decrypt if a specific event occurs. Specific constructions of TRE can be easily extended to support such a notion by releasing an event-specific trapdoor after the event occurred.

It is stressed in [6] that TRE is different from TCE, but the applicabilities in different scenarios are not elaborated. In certain scenarios, we note that TCE indeed provides a partial solution to the TRE problem.

2.1 TRE from Double Encryption

Two parties' secret knowledge are required to decrypt, it is natural to ask "why not double encrypt?". In the TRE scheme proposed by Di Crescenzo *et al.* [10], the ciphertext is $c = \mathsf{Enc}_{ek_A}(\mathsf{Enc}_{ek_R}(m))$ where m is the message, ek_A and ek_R is the public key of the agent and the receiver respectively, and $\mathsf{Enc}_{ek}(m)$ denotes the public key encryption of message m under the public key ek. This ciphertext is sent together with the release-time to the receiver. On receiving c and the time information, the receiver interacts with the agent using a "conditional oblivious transfer protocol". Such a protocol ensures that if the release-time is not less than the current time, it decrypts the first level and gives $\mathsf{PKE.Enc}_{ek_R}(m)$; otherwise, receiver learns nothing. However, this protocol [10] is computationally intensive. Moreover, chosen-ciphertext security of double encryption [13] is not considered.

2.2 Privacy Issues

The key difference between TCE and TRE is that the sender must contact the SEM (once) for delegating the token in the case for TCE. It is true that the solution is interactive, but the communication is quite minimal.

[1] Time-lock puzzle approach (e.g. [19]), in which the receiver has to invest a significant computational effort to solve a difficult problem, is not considered in this paper. This approach does not involve a mediator but it is computationally expensive for the receiver and the release-time is not precisely controllable.

The senders in TCE cannot be anonymous to the SEM, but in our motivating scenarios, the SEM is going to charge the senders for the service fee anyway. Moreover, it is not an issue if the senders are performing the role of SEM themselves (in the applications we will discuss shortly afterward). Another privacy issue is about release-time (or release-condition) confidentiality, i.e. the ciphertext itself does not leak any information about the pre-specified release condition. Similar to event-release encryption, the SEM knows the event's details in order to release the token at the right time[2]. We will initiate the study of this concept in the context of TCE. Apart from using the event's details, malicious SEMs may try to find out what ciphertexts are controlled by the token they hold. This concern will be addressed by our proposed notion of ciphertext unlinkability.

2.3 Pre-open Capability

In TRE, a single time-specific trapdoor controls the decryption of all messages of the whole system to be recovered at a given time. Pre-opening of some messages in a general TRE scheme means pre-opening of all messages at a given time.

In [17], the concept of pre-open capability is introduced, such that the sender can help the receiver to decrypt the ciphertext (without sending the stored plaintext) by publishing a pre-open key. As argued in [11,17], many applications of TRE (e.g. electronic auction) potentially needs the pre-open capability. However, it is later noted in [11] that the security model and hence the scheme in [17] does not cover a realistic attack; namely, the sender can make the receiver to get a message different from which was originally encrypted by sending a false pre-open key. A concrete scheme secure against this attack is also proposed in [11]. This security requirement matches the strong existential token unforgeability of TCE. In other words, any TCE scheme with strong existential token unforgeability automatically supports pre-open capability (in the sense of [11]). On the other hand, this capability may not exist in a general TRE scheme like [6].

2.4 Requirements on the Security-Mediator

Another minor difference is that the SEM must retain all the tokens collected instead of using a system's private key to generate all time-specific trapdoors. Moreover, the fact that each sender prepares his/her own token independently implies that SEM may need to release more than one token at a time; in contrast to a single time-specific trapdoor in TRE. Firstly, note that a single token can control many ciphertexts for TCE. Besides, the trusted agent in the TRE paradigm is usually a powerful server with dedicated protection, since its compromise means the whole system's security is broken. On the other hand, TCE enjoys the advantage that the leakage of one token does not affect the encryption using another independent token. The trust level in the TRE paradigm is similar to that of certificate authority of a public key infrastructure while in TCE the role of the SEM can be performed by lawyers or some financial institutions.

[2] If the same token is controlling the ciphertexts of more than one recipient, the SEM needs to know whom the token should be released to instead of just making it public.

2.5 Security and Efficiency

To the best of the author's knowledge, there is no TRE scheme in the standard model. Existing TRE schemes [5,6,17,18] are often constructed by a combination of an unlock method made possible by identity based encryption (IBE) with a public key encryption (PKE). However, IBE is in general more computationally expensive than PKE. In particular, existing TRE schemes involves pairing computation which is computationally expensive (e.g. see [1] for a brief discussion). On the other hand, our implementation of TCE is pairing-free. One should employ TCE instead of TRE whenever possible, for better security and efficiency.

2.6 More Applications

After knowing the differences, we see many more applications of TCE. Generally speaking, TCE is applicable in scenarios where the receiver believes that the senders will not prepare incorrect messages for encryption, but does not want the senders to change their mind once the messages are sent.

Rapid Dissemination of Information. Scheduled payment is one of the motivating applications discussed in the seminal work [2]. Suppose a company wants to send pay slips to its employees, in a way that each employee is required to perform decryption to get an authorization code for obtaining the salary. For obvious reason the company may not want to pay before the payday. On the other hand, sending pay slips on a single day is undesirable due to the large number of employees (which causes a large volume of network traffic).

The same situation applies for rapid dissemination of freshly published, crucial or highly priced, and formerly secret information [18], such as stock market values, strategic business plans, news agencies timed publications, licensed software updates, etc. Considering the size of the confidential information is significantly larger than the size of an encryption key (which is often the case), timely distribution of information to a large and distributed community of users is often impeded by network traffic jams. With TCE, messages (the pay slips, or any confidential material) can be encrypted beforehand and gradually distributed. Holding the ciphertext without the token means no one can perform the decryption. At a later time (the payday, or when some disclosure requirement is satisfied), a single small-sized token can be released to enable all the recipients to read the previously encrypted messages.

Sealed-bid Auctions and Electronic Lotteries. Sealed-bid auction is naturally supported by TCE. All bidders can encrypt their bids using TCE and send the resulting ciphertexts to the dealer, who will "publish" all the encrypted bids in some bulletin board where all bidders can access. At the end-time of the auction (or when all bidders agree to open their bids), all bidders publish their tokens. Due to the strong existential token unforgeability (to be defined formally), they cannot come up with a token such that the ciphertext decrypts to a message different from the one that is encrypted before. On the other hand, no one gain any knowledge before the release time of the tokens.

Electronic lotteries can be realized similarly[3]. The decryption token and the encrypted ticket can be used as a proof for the ownership of a winning ticket.

3 Preliminaries

If $\mathcal{S}$ is a finite set, $x \leftarrow \mathcal{S}$ denotes the operation of picking an element at random and uniformly from S. If S is a probabilistic algorithm, $x \leftarrow \mathsf{S}(y)$ denotes S has been executed on some specified input y and its random output has been assigned to the variable x. If $\ell \in \mathbb{N}$ then 1^ℓ denotes the bit-string of ℓ ones. The term "negligible" refers to the class of negligible functions in the parameter of ℓ. The acronym PPT stands for probabilistic polynomial time. Finally, let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two probability distributions over the same base set $\mathcal{S}$; the statistical difference between $\mathcal{D}_1$ and $\mathcal{D}_2$, is defined as $||\mathcal{D}_1 - \mathcal{D}_2|| = \sum_{s \in \mathcal{S}} |\Pr_{\mathcal{D}_1}[s] - \Pr_{\mathcal{D}_2}[s]|$.

3.1 Trapdoor Partial One-Way Functions

A trapdoor one-way function is a function that is easy to compute, but difficult to invert completely without an extra piece of information termed as trapdoor. A trapdoor partial one-way function (TPOWF) [14] is one that inversion means only part of the input can be recovered. TPOWF is used in the generic construction of TCE in [15] which we will review.

Formally, TPOWF refers to a family of injective maps $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are polynomial size set families, with the following three PPT algorithms.

1. $\mathsf{Gen}(1^\ell)$ outputs a key pair (ek, dk) and the descriptions of the sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$.
2. Eval is a deterministic evaluation algorithm that takes on input ek, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and outputs $f_{ek}(x, y)$.
3. Inv is a deterministic inversion algorithm that takes on input dk and $z \in \mathcal{Z}$, returns $x \in \mathcal{X}$ such that $z = f_{ek}(x, y)$ where $y \in \mathcal{Y}$.

An element from $\mathcal{X}$ can be samplable and recognizable in polynomial time.

The partial one-wayness requires the following probability to be negligible, for any PPT adversary $\mathcal{A}$ without the trapdoor key dk:

$$\Pr\left[\begin{array}{c}(ek, dk, \mathcal{X}, \mathcal{Y}, \mathcal{Z}) \leftarrow \mathsf{Gen}(1^\ell); \\ x \leftarrow \mathcal{X}; y \leftarrow \mathcal{Y} : \mathcal{A}(ek, f_{ek}(x, y)) = x\end{array}\right].$$

3.2 Secret Key Encryption

Our construction relies on a stateless secret key encryption (SKE) scheme, which is defined by a triple of PPT algorithm ($\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$):

– Gen takes security parameter 1^ℓ; outputs a secret key $sk \in \mathcal{K}_\ell$, the descriptions of a message space $\mathcal{M}_\ell$ and the secret key space $\mathcal{K}_\ell$.

[3] We claim that TCE is just a helper tool in these scenarios. There are other specific security requirements (e.g. see [8]) which TCE may not help.

– Enc (probabilistic) takes sk and a message m, outputs a ciphertext C.
– Dec (deterministic) takes sk and C, outputs a message m or $\perp$ if C is invalid.

For correctness, we require $\mathsf{Dec}_{sk}(\mathsf{Enc}_{sk}(m)) = m$ for all $\ell \in \mathbb{N}$, all $sk \in \mathcal{K}_\ell$ generated by $\mathsf{Gen}(1^\ell)$ and for all message $m \in \mathcal{M}_\ell$. For security, we require indistinguishability against adaptive chosen-plaintext attack (CPA). Formally speaking, the following is negligible for all $\mathsf{PPT}\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$$\left| \Pr \left[\begin{array}{rcl} (sk, \mathcal{M}_\ell, \mathcal{K}_\ell) & \leftarrow & \mathsf{Gen}(1^\ell); \\ (m_0, m_1, \beta) & \leftarrow & \mathcal{A}_1^{\mathsf{Enc}_{sk}(\cdot)}(1^\ell); \\ \tilde{c} & \leftarrow & \mathsf{Enc}_{sk}(m_b); \\ \tilde{b} & \leftarrow & \mathcal{A}_2^{\mathsf{Enc}_{sk}(\cdot)}(\tilde{c}, \beta); \end{array} : b = \tilde{b} \right] - \frac{1}{2} \right|.$$

3.3 Public Key Encryption

Our construction is built on a public key encryption scheme (PKE), which is defined by a triple of PPT algorithm $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

– Gen takes security parameter 1^ℓ; outputs a key pair (ek, dk), the descriptions of a message space $\mathcal{M}_{ek}$ and a ciphertext space $\mathcal{C}_{ek}$.
– Enc is a randomized algorithm takes an encryption key ek, and a message m as input, outputs a ciphertext C.
– Dec is a deterministic algorithm takes a decryption key dk, and a ciphertext C, outputs a message m or $\perp$ if C is invalid.

For correctness, we require $\mathsf{Dec}_{dk}(\mathsf{Enc}_{ek}(m)) = m$ for all $\ell \in \mathbb{N}$, (ek, dk) given by $\mathsf{Gen}(1^\ell)$ and all $m \in \mathcal{M}_{ek}$. For security, we require indistinguishability against adaptive chosen-ciphertext attack (CCA) for multiple-user[4]. Formally speaking:

$$\left| \Pr \left[\begin{array}{rcl} (ek_i, dk_i, \mathcal{M}_{ek_i}, \mathcal{C}_{ek_i}) & \leftarrow & \mathsf{Gen}(1^\ell), \forall i \in \{1, \cdots, n\}; \\ (m_0, m_1, \beta) & \leftarrow & \mathcal{A}_1^{\mathsf{Dec}^{\perp}_{dk_1}(\cdot), \cdots, \mathsf{Dec}^{\perp}_{dk_n}(\cdot)}(1^\ell); \\ \tilde{c_i} & \leftarrow & \mathsf{Enc}_{ek_i}(m_b), \forall i \in \{1, \cdots, n\}; \\ \tilde{b} & \leftarrow & \mathcal{A}_2^{\mathsf{Dec}^{\tilde{c}_1}_{dk_1}(\cdot), \cdots, \mathsf{Dec}^{\tilde{c}_n}_{dk_n}(\cdot)}(\tilde{c}, \beta); \end{array} : (b = \tilde{b}) \right] - \frac{1}{2} \right|$$

is negligible for all $\mathsf{PPT}\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where the decryption oracle $\mathsf{Dec}^{\tilde{c}}_{dk}(c)$ is:

$$\text{if } (c = \tilde{c}) \text{ then aborts; else return } \mathsf{Dec}_{dk}(c).$$

3.4 Commitment

The final building block of our construction is a commitment scheme (COM), which is a pair of PPT algorithms $(\mathsf{Send}, \mathsf{Rec})$ such that:

– Send takes as input 1^ℓ and a message $r \in \{0,1\}^*$; returns $(com, dec) \in \mathcal{U} \times \mathcal{V}$ representing the commitment and the decommitment string respectively.
– Rec takes $(1^\ell, com, dec)$ and returns $r \in \{0,1\}^* \cup \{\perp\}$.

[4] This is weaker than the CCA-security in [3] since we only provide the encryption under different public keys of the same message instead of a left-or-right oracle [3].

Correctness requires $\mathsf{Rec}(1^\ell, \mathsf{Send}(1^\ell, r)) = r, \forall \ell \in \mathbb{N}, r \in \{0,1\}^*$. For security, we consider binding and hiding properties. Hiding means that *com* should not reveal information about r; binding means an adversarially generated *com* can be "opened" to only a single legal value of r. The formal definitions are as follows.

- **Hiding:** $\forall r_1, r_2 \in \{0,1\}^*, ||C_\ell(r_1) - C_\ell(r_2)|| = O(2^{-\ell})$, where $C_\ell(r)$ denotes the distribution over the commitment strings for r, i.e. the first component of $\mathsf{Send}(1^\ell, r)$'s output.
- **Binding:** For all PPT adversaries $\mathcal{A}$, the following probability is negligible:

$$\Pr\left[\begin{array}{c}(com, dec, dec') \leftarrow \mathcal{A}(1^\ell) : \\ (\mathsf{Rec}(1^\ell, com, dec) \neq \perp) \wedge \\ (\mathsf{Rec}(1^\ell, com, dec') \neq \perp) \wedge \\ (\mathsf{Rec}(1^\ell, com, dec) \neq \mathsf{Rec}(1^\ell, com, dec'))\end{array}\right].$$

4 Definitions of Token-Controlled Public Key Encryption

A token-controlled public key encryption consists of the below PPT algorithms:

- Gen takes security parameter 1^ℓ; outputs a key pair (ek, dk), the descriptions of a message space $\mathcal{M}_{ek}$, a ciphertext space $\mathcal{C}_{ek}$, an encrypting token space and a delegated token space.
- Tok takes security parameter 1^ℓ, returns an encrypting token τ, a token verifier *com* to be used in Enc, and a delegated token *dec* for the SEM.
- Enc is a randomized algorithm taking an encryption key ek, a message m (with an optional auxiliary message aux,) an encrypting token τ, and a token verifier *com* as input, outputs a ciphertext C.
- PDec is a deterministic algorithm taking a decryption key dk and a ciphertext C as input, outputs either an auxiliary message aux, or $\perp$ if C is invalid.
- Dec is a deterministic algorithm taking a decryption key dk, a ciphertext C and a delegated token *dec* as input, outputs either a message m, $\perp_t$ if *dec* is not valid, or $\perp_c$ if C is invalid.

For correctness, we require (aux can be an empty string)

- $\mathsf{PDec}_{dk}(\mathsf{Enc}_{ek}(m, aux, \tau, com)) = aux$ for all $\ell \in \mathbb{N}$, (ek, dk) given by $\mathsf{Gen}(1^\ell)$, all $m, aux \in \mathcal{M}_{ek}$ and all (τ, com, dec) given by $\mathsf{Tok}(1^\ell)$;
- $\mathsf{Dec}_{dk}(\mathsf{Enc}_{ek}(m, aux, \tau, com), dec) = m$ for all $\ell \in \mathbb{N}$, (ek, dk) given by $\mathsf{Gen}(1^\ell)$, all $m, aux \in \mathcal{M}_{ek}$ and all (τ, com, dec) given by $\mathsf{Tok}(1^\ell)$.

We generalized the notion in [2,15] a bit, the differences are highlighted below.

- An auxiliary message (which may contain the condition under which the recipient can get the delegated token) and the corresponding PDec algorithm to recover such message from the ciphertext by the decryption key, are added.
- The token used by the sender in encrypting the message (the encrypting token) is not necessary equal to the token delegated from the sender to the SEM (the delegated token) for future decryption.
- A token verifier is passed from TCE.Tok to TCE.Enc.
- Decryption algorithm gives different signals for different failure cases, depending on whether the delegated token or the ciphertext is invalid.

4.1 Relations Among Existing Notions

The end-users of TCE are those who hold the private keys. Attacks model of [2] and [15] make the distinctions between insider and outsider. The former has the private key and the later does not. The original definition in [2] further distinguishes between two types of outsider. Type I attacker does not have the token while Type II attacker models a SEM since it is supplied with the token. Counting also the insider attacker, this makes three kinds of attacker altogether.

It seems that Type II attacker is strictly stronger than Type I, which is also suggested by [15]. However, not to forget the oracle queries available to these two types of attackers. A decryption oracle in the model of [2] and [15] takes a token-ciphertext pair as input and decrypts the given ciphertext using the supplied token and the private key embedded. If a Type II attacker is allowed to query the decryption oracle for the challenge ciphertext, the challenge can be solved trivially since it *knows* the *correct* token. On the other hand, a Type I attacker is not supplied with the correct token, such a restriction is not necessary.

Nonetheless, it is easy to see that with the private key, a decryption oracle without any restriction can be easily realized; while a Type I attacker only has access to decryption oracle but not the private key itself. In other words, an *insider* who possesses the private key is strictly stronger than a Type I attacker. Due to these observations, we still "follow" the model of Galindo and Herranz [15] in the sense that we only consider two kinds of attacker: outsider and insider.

4.2 Outsider Indistinguishability

Outsider indistinguishability captures the situation where the attacker (e.g. the SEM) has the token but without the corresponding private key. This property also captures the reusability of the same token. If the same token is used to encrypt messages to many different receivers, and at a later stage some of these receivers obtain the correct token, they cannot obtain any partial information about message encrypted to some different receivers.

We give two extra level protections considered first time in this work.

1. The scheme should be secure for adversarially chosen token, which provides security when the message sender is tricked to use some token, for example by malicious code introduced to the machine where the ciphertext is prepared.
2. Partial decryption oracle is assumed for the adversary. An intention of introducing this oracle is somewhat similar to the role of the decryption oracle in the standard CCA attack for traditional PKE. It is possible that the attacker can somehow access the machine of the subject under attack, that can compute (part of the) decryption result for the attacker. Moreover, user may performs the partial decryption well before the recipient of the token, and carelessly leaves the partial decryption result unprotected in the belief that it is secure to do so as long as no one knows the token. This gives another motivation for the partial decryption oracle.

Outsider indistinguishability is formally modeled by the following game.

1. **Initialization:** The challenger $\mathcal{C}$ executes $\mathsf{TCE.Gen}$ for n times independently, i.e. $(ek_i, dk_i) \leftarrow \mathsf{TCE.Gen}(1^\ell)$ for $i = 1, \cdots, n$. The adversary $\mathcal{A}$ gets the public keys $\langle ek_1, \cdots, ek_n \rangle$, but not the decryption keys $\langle dk_1, \cdots, dk_n \rangle$.
2. **Phase 1:** The challenger $\mathcal{C}$ entertains any queries to token generation, encryption, partial decryption and complete decryption oracle from $\mathcal{A}$. Important points about the queries include:
   (a) For encryption oracle, $\mathcal{A}$ can supply the encrypting token and the token verifier, but it is not required to hand in the delegated token.
   (b) For partial decryption oracle, $\mathcal{A}$ does not required supplying the delegated token *dec*, and the oracle returns a partial decryption result by dk_i, that can possibly be further decrypted by the delegated token.
   (c) For complete decryption oracle, $\mathcal{A}$ should supply the delegated token *dec*, but it is not required to hand in the decryption key dk_i.
3. **Challenge:** $\mathcal{A}$ outputs two messages m_0, m_1 from $\cap_{1 \leq i \leq n} \mathcal{M}_{pk_i}$ of the same length, an encrypting token τ and a token verifier *com* to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit b and returns $c_i^* = \mathsf{TCE.Enc}_{pk_i}(m_b, \tau, com)$ for $i = 1, \cdots, n$.
4. **Phase 2:** The adversary proceeds as in Phase 1, with two restrictions:
   (a) $\mathcal{A}$ cannot ask for the partial decryption of a ciphertext c_i^* under decryption key dk_i for any $i = 1, \cdots, n$; and
   (b) $\mathcal{A}$ cannot ask for the complete decryption of a ciphertext c_i^* under decryption key dk_i for any $i = 1, \cdots, n$, irrespective of what the delegated token it supplies[5].
5. **Guess:** Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

The adversary's advantage is defined as $|\Pr[b' = b] - \frac{1}{2}|$.

4.3 Insider Indistinguishability

Insider security of TCE basically means the ciphertext is indistinguishable even with the secret key, as long as the token is unknown. Moreover, an embedded-token encryption oracle is provided to the adversary. The intuitive meaning of this oracle is to ensure the ciphertext would not leak any information about the token that is useful for distinguishing the ciphertexts.

This property is formally modeled by the following game.

1. **Initialization:** The challenger $\mathcal{C}$ executes $\mathsf{TCE.Tok}$, i.e. $(\tau, com, dec) \leftarrow \mathsf{TCE.Tok}(1^\ell)$. Both τ and *dec* are kept secret, whereas the security parameter 1^ℓ and the token verifier *com* are given to $\mathcal{A}$.

[5] It does not make sense to entertain the complete decryption of the challenge ciphertext. Suppose the delegated token is the same as the one implicitly defined by the encrypting token and the token verifier chosen by the adversary in the preparation of the challenge ciphertext, decrypting for the adversary means no challenge is given to the adversary at all. On the other hand, if the delegated token is a different one, from the strong existential token unforgeability we know that the decryption either returns the original message or invalid. There is no challenge for the former case.

2. **Phase 1:** $\mathcal{A}$ can make queries to an embedded-token encryption oracle $\mathsf{TCE.Enc}_{ek}(m, \tau, com)$, for public key ek and message m of its choice.
3. **Challenge:** $\mathcal{A}$ outputs a challenge public key ek, and two messages $m_0, m_1 \in \mathcal{M}_{ek}$ of the same length to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and returns $c^* = \mathsf{TCE.Enc}_{ek}(m_b, \tau, com)$.
4. **Phase 2:** The adversary can proceed as in Phase 1, without any restriction.
5. **Guess:** $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

The adversary's advantage is defined as $|\Pr[b' = b] - \frac{1}{2}|$.

Up to this point, it is the definition from [15]. It is possible to consider some stronger notions.

1. Instead of returning a single token verifier *com* to $\mathcal{A}$, it is possible to return polynomially-many (says n) of them and ask $\mathcal{A}$ to choose one to be challenged with. We consider the basic notion since a scheme secure in the "single-token sense" is also secure in the "multiple-token sense" with loss of a factor of n for the tightness of the security reduction.
2. An embedded-token *decryption* oracle can be provided to the adversary, with the natural restriction that it cannot be used to decrypt the challenge ciphertext. We term this notion as *strong* insider indistinguishability, which is also firstly considered in our work. But we made the choice of not including this in our standard definition since we believe that the SEM is paid to well-protect the secret token from arbitrary access in realistic scenarios like the wills encryption. Nevertheless, we will discuss how one can extend our proposed scheme to achieve it in Section 7.

4.4 Strong Existential Token Unforgeability

This is the property proposed in [15] which does not supported by the original schemes in [2]. Informally, this new notion ensures the infeasibility to come up with a valid ciphertext c such that it can be decrypted to two different messages under two different tokens. The adjective "strong" here means not only a fake token but also the ciphertext are forged by the adversary.

This property is formally modeled by the following game [15].

1. **Initialization:** The challenger $\mathcal{C}$ executes $\mathsf{TCE.Gen}$, i.e. $(ek, dk) \leftarrow \mathsf{TCE.Gen}(1^\ell)$ The adversary $\mathcal{A}$ receives the public key ek, but not the decryption key dk.
2. **Oracle Access:** $\mathcal{A}$ can make queries like **Phase 1** in the insider indistinguishability game.
3. **Forgery:** $\mathcal{A}$ outputs two delegated tokens dec and dec', and a ciphertext C.

The adversary's advantage is defined as

$$\Pr\left[\begin{array}{c}(\mathsf{TCE.Dec}_{(dk,dec)}(C) = m) \wedge (\mathsf{TCE.Dec}_{(dk,dec')}(C) = m') \\ \wedge\ (dec \neq dec') \wedge (m \neq m') \wedge (m \neq \perp) \wedge (m' \neq \perp)\end{array}\right]$$

Similar to the discussion of insider indistinguishability, we can easily get unforgeability for n public keys from a scheme that is unforgeable for a single public key with loss of a multiplicative factor n in the tightness of the security reduction.

4.5 Release-Condition Confidentiality and Ciphertext Unlinkability

While in theory the ciphertext of TCE may not include any information about the release condition, this information should be sent to the recipient in practice. On the other hand, it is undesirable for any other people (including the SEM) to know about the associated release condition by just inspecting the ciphertext. This is release-condition confidentiality. Apart from the knowledge of the release-condition, a malicious SEM may try to use the knowledge of the delegated token to find out which ciphertext is controlled by the token. We want to prevent this from happening with ciphertext unlinkability.

Due to the lack of space, we just describe how to change the notion of outsider indistinguishability to model these requirements. Release-condition confidentiality can be modeled easily by asking the adversary to return a single message to be encrypted and a pair of release conditions. A ciphertext encrypting the message with a random condition among the pair is returned to the adversary. The adversary's advantage is defined as the probability of telling the random choice the challenger has made over $\frac{1}{2}$. For ciphertext unlinkability, the adversary chooses a message m, two pairs of encrypting token and token verifier (τ_0, com_0) and (τ_1, com_1) to be challenged with. The adversary's advantage is defined as the probability of guessing correctly the bit b in $\mathsf{TCE.Enc}_{pk_i}(m, \tau_b, com_b)$ over $\frac{1}{2}$.

5 Analysis of an Existing Generic Construction

We show the generic construction of TCE in [15] is insecure against insider attack. Our analysis applies for all TPOWF, e.g. the RSA function suggested in [15].

5.1 Review

$\mathsf{TCE.Gen}^{GH}$: The input is a security parameter 1^ℓ.

1. Let f be a TPOWF family over the sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ as previously defined;
2. Run $(ek, dk) \leftarrow \mathsf{TPOWF.Gen}^f(1^\ell)$;
3. Let the message space and the token space be $\mathcal{M}_\ell = \{0,1\}^{p(\ell)}$ and $\mathcal{T}_\ell = \{0,1\}^{t(\ell)}$ for some polynomial $p(\ell)$ and $t(\ell)$ respectively;
4. Let $\tilde{G} : \mathcal{X} \times \mathcal{T}_\ell \rightarrow \mathcal{M}_\ell$ and $\tilde{H} : \mathcal{X} \times \mathcal{M}_\ell \rightarrow \mathcal{Y}$ be two hash functions, the ciphertext space $\mathcal{C}_{ek}$ is $\mathcal{Z} \times \mathcal{M}_\ell$;
5. The public parameter is $\langle ek, \tilde{G}, \tilde{H}, \mathcal{M}_\ell, \mathcal{T}_\ell, \mathcal{C}_{ek} \rangle$ and the decryption key is dk.

$\mathsf{TCE.Tok}^{GH}$: Output a random element from τ from $\mathcal{T}_\ell$. This token serves as both the encrypting token (will be used in $\mathsf{TCE.Enc}^{GH}$) and the delegated token (will be used in $\mathsf{TCE.Dec}^{GH}$) in our new framework.

$\mathsf{TCE.Enc}^{GH}$: Suppose the message to be encrypted is $m \in \mathcal{M}_\ell$ and the token chosen is τ, the ciphertext is $c = \langle f_{ek}(x, y), \tilde{G}(x, \tau) \oplus m \rangle$, where $y = \tilde{H}(x, m)$ and x is uniformly chosen in $\mathcal{X}$.

$\mathsf{TCE.Dec}^{GH}$: Suppose the ciphertext to be decrypted is $\langle c_1, c_2 \rangle$.

1. Compute $x = \mathsf{TPOWF.Inv}^f(c_1)$;
2. Compute $m = \tilde{G}(x, \tau) \oplus c_2$;
3. Return m if $c_1 = f_{ek}(x, \tilde{H}(x, m))$;
4. Return $\perp$ if the above equality does not hold, or either one of $\mathsf{TPOWF.Inv}^f(c_1)$ and $\tilde{G}(x, \tau)$ cannot be computed.

5.2 Insecurity Against Insider Attack in the Old Model

We start by pointing out the problematic statement in the proof of insider security in [15]. It is claimed that the value b is independent from $\mathcal{A}$'s view as long as $\mathcal{A}$ does not query $\tilde{G}$ at the point (x^*, τ^*), where b is the random bit chosen by the challenger, x^* and τ^* are the value of x and the token associated with the challenge ciphertext respectively. The statement is not true since the first part of the ciphertext component is $f_{ek}(x^*, y^*)$, where $y^* = \tilde{H}(x^*, m_b)$. In other words, the value of b is dependent on the adversary's view (part of the challenge ciphertext) no matter whether a "clever" $\tilde{G}$ query is made.

Suppose the adversary got $c^* = \langle c_1, c_2 \rangle = \langle f_{ek}(x^*, y^*), \tilde{G}(x^*, \tau^*) \oplus m_b \rangle$ as the challenge ciphertext, a concrete attack is as follows.

1. Compute $x = \mathsf{TPOWF.Inv}^f(c_1)$ with the decryption key dk;
2. For $b' \in \{0, 1\}$, check whether $c_1 = f_{ek}(x, \tilde{H}(x, m_{b'}))$.
3. Output b' that makes the above equality holds.

The function f is injective, the adversary must get $x = x^*$. Being a function, f must be deterministic, and the value of c_1 will be either $f_{ek}(x, \tilde{H}(x, m_0))$ or $f_{ek}(x, \tilde{H}(x, m_1))$. Note that no $\tilde{G}$ query is made and the token is not required.

One may try to fix the scheme by setting $y = \tilde{H}(x, \tau, m)$ (or many possibilities of inputs). Unfortunately, we will show that their scheme is insecure against outsider in our new model.

5.3 Insecurity Against Outsider Attack in the New Model

Our notion of outsider indistinguishability gives the attacker accesses of a partial decryption oracle using user's decryption key. Note that the integrity checking (the checking step determines whether a message m or an invalid symbol $\perp$ is returned) in their scheme requires the token as an input and cannot be put into the partial decryption algorithm. The use of a trapdoor partial one-way function without any integrity checking is surely not enough for CCA-security.

Suppose RSA is used to instantiate TPOWF, with n, e, d as the public modulus, the encryption exponent and the decryption exponent respectively, such that $ed = 1 \bmod \phi(n)$ where $\phi(\cdot)$ is the Euler totient function. The partial decryption oracle computes the d-th power for the attacker. The first component of the challenge ciphertext contains $(x^e) \bmod n$, where x is the value associated with the challenge ciphertext. An adversary can just find a random $r \in \mathbb{Z}_n$, give $(r^e)(x^e) \bmod n$ (i.e. a ciphertext different from the challenge one) to the partial decryption oracle, get back $x' = rx$, and obtain the message by $c_2 \oplus \tilde{G}(x' r^{-1}, \tau)$.

6 A New Generic Construction in the Standard Model

We assume that PKE encrypts message that is concatenated from the ciphertext of SKE, the commitment string of COM, and the auxiliary information.

TCE.Gen: The input is a security parameter 1^ℓ.

1. Run $(ek, dk, \mathcal{M}_{ek}, \mathcal{C}_{ek}) \leftarrow \mathsf{PKE.Gen}(1^\ell)$;
2. The message space is the message space $\mathcal{M}_\ell$ of SKE while the ciphertext space is $\mathcal{C}_{ek}$. The encrypting token space is the key space $\mathcal{K}_\ell$ of SKE, the token verifier space is the commitment string space $\mathcal{U}_\ell$ of COM, the delegated token space is the decommitment space $\mathcal{V}_\ell$ of COM;
3. The public parameter is $\langle ek, \mathcal{M}_\ell, \mathcal{K}_\ell, \mathcal{C}_{ek}, \mathcal{U}_\ell, \mathcal{V}_\ell \rangle$;
4. The secret key is dk.

TCE.Tok: This algorithm just needs a random coin.

1. Randomly selects $\tau \in \mathcal{K}_\ell$;
2. Run $(com, dec) \leftarrow \mathsf{Send}(1^\ell, \tau)$;
3. Output (τ, com, dec) as the encrypting token, the token verifier, and the delegated token respectively.

TCE.Enc: Taking a message $m \in \mathcal{M}_\ell$, an optional auxiliary message $aux \in \mathcal{M}_\ell$, an encrypting token τ and a token verifier com, output $c = \mathsf{PKE.Enc}_{ek}(\sigma||com||aux)$, where $\sigma \leftarrow \mathsf{SKE.Enc}_\tau(m)$.

TCE.PDec: Suppose the ciphertext is c'.

1. Run $(\sigma'||com'||aux) \leftarrow \mathsf{PKE.Dec}_{dk}(c')$;
2. Output aux or return $\perp$ if step 1 fails.

TCE.Dec: Suppose the ciphertext is c' and the delegated token is dec'.

1. Run $(\sigma'||com'||aux) \leftarrow \mathsf{PKE.Dec}_{dk}(c')$;
2. Run $\tau' \leftarrow \mathsf{Rec}(1^\ell, com', dec')$;
3. Output $m' \leftarrow \mathsf{SKE.Dec}_{\tau'}(\sigma)$;
4. If step 2 returns $\perp$, return $\perp_t$; else if step 1 or step 3 returns $\perp$, return $\perp_c$.

Correctness of the scheme is easy to check: dk is used in decryption to get σ' and com'; with (com', dec'), τ' can be recovered, which is used to undo the secret key encryption, i.e. decrypt σ' to get back the original message m'.

6.1 Outsider Attacks

If there exists an PPT adversary $\mathcal{A}_1$ that breaks the outsider indistinguishability of our TCE scheme, we show how to build an algorithm $\mathcal{B}_1$ that makes use of $\mathcal{A}_1$ to break the indistinguishability of PKE in the multi-user setting.

1. **Initialization:** Let $\mathcal{C}_1$ denotes $\mathcal{B}_1$'s challenger in breaking the indistinguishability of PKE. $\mathcal{B}_1$ obtains the list of public keys $\langle ek_1, ek_2, \cdots, ek_n \rangle$ and the list of ciphertext space $\langle \mathcal{C}_{ek_1}, \cdots, \mathcal{C}_{ek_n} \rangle$ from $\mathcal{C}_1$. Using the same security parameter ℓ chosen by $\mathcal{C}_1$, the commitment space $\mathcal{U}_\ell$ and the decommitment space $\mathcal{V}_\ell$ are defined accordingly; and SKE.Gen is executed to obtain $sk \in \mathcal{K}_\ell$, the message space $\mathcal{M}_\ell$ and the secret key space $\mathcal{K}_\ell$. The list of TCE public parameters $\langle ek_i, \mathcal{M}_\ell, \mathcal{K}_\ell, \mathcal{C}_{ek_i}, \mathcal{U}_\ell, \mathcal{V}_\ell \rangle$ for $i \in \{1, \cdots, n\}$ are given to $\mathcal{A}_1$.
2. **Phase 1:**
   (a) Token generation oracle can be easily simulated.
   (b) Encryption oracle can be simulated with the help of sk.
   (c) Partial decryption oracle can be simulated by PKE's decryption oracle.
   (d) Complete decryption oracle is easy to simulate since the adversary is required to supply the delegated token.
3. **Challenge:** The adversary outputs two equal length messages $m_0, m_1 \in \cap_{1 \le i \le n} \mathcal{M}_{pk_i}$, an encrypting token τ' and a token verifier com' to $\mathcal{B}_1$. $\mathcal{B}_1$ then computes $m'_0 \leftarrow \mathsf{SKE.Enc}_{\tau'}(m_0)$ and $m'_1 \leftarrow \mathsf{SKE.Enc}_{\tau'}(m_1)$. The message pair to be passed to $\mathcal{C}_1$ is $(m'_0||com', m'_1||com')$. $\mathcal{B}_1$ then forwards the challenge from $\mathcal{C}_1$ to $\mathcal{A}_1$.
4. **Phase 2:** $\mathcal{A}_1$ asks queries as in Phase 1. For partial/complete decryption query $\mathcal{B}_1$ needs to ask the underlying decryption oracle. With the restrictions specified in the definition in Section 4, $\mathcal{B}$ would not pass $\mathcal{C}_1$'s challenge back to $\mathcal{C}_1$'s decryption oracle, so $\mathcal{B}_1$ can simulate all the decryption queries faithfully.
5. **Guess:** $\mathcal{A}_1$ outputs a guess $b' \in \{0, 1\}$, b' is output by $\mathcal{B}_1$ as its final guess.
6. **Analysis:** It is easy to see that the advantage of $\mathcal{B}_1$ in breaking the indistinguishability of PKE in the multi-user setting is the same as the advantage of $\mathcal{A}_1$. With [3, Theorem 1], the advantage of $\mathcal{B}_1$ in breaking the indistinguishability of PKE for single-user is ϵ/n.

It is easy to see that release-condition confidentiality and ciphertext unlinkability can be proven in a similar way.

6.2 Insider Indistinguishability

We show the probability of breaking the insider indistinguishability of our scheme is negligible, given that COM is hiding and SKE is CPA-secure. Our proof follows the structural approach advocated by Shoup [20] in defining a sequence of games.

We use the notation $\Pr_i[\cdot]$ to denote the probability of an event occurring in Game i. Game 0 is the original game in which the adversary attacks our scheme. We are going to upper-bound $|\Pr_0[\mathsf{Succ}] - \frac{1}{2}|$, where Succ denotes the event that the adversary's output bit b' matches the bit b chosen in the insider indistinguishability game.

In Game 1, the simulator chooses another τ' from $\mathcal{K}_\ell$ and use it in the embedded-token encryption oracle, but still outputs com from $(com, dec) \leftarrow$

$\mathsf{Send}(1^\ell, \tau)$ instead of $\mathsf{Send}(1^\ell, \tau')$. Let NoHide denote the event the hiding property of COM is broken, we have $|\Pr_0[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]| \leq \Pr_1[\mathsf{NoHide}]$ since Game 0 and Game 1 are identical until NoHide occurs.

Game 2 is just a bridging step. We simulate the embedded-token encryption oracle with the underlying encryption oracle of SKE but not τ'. Nevertheless, we can view the encryption oracle of SKE is using some random key following the same distribution as τ'. We have $\Pr_2[\mathsf{Succ}] = \Pr_1[\mathsf{Succ}]$ and $|\Pr_2[\mathsf{Succ}]| = \epsilon + \frac{1}{2}$, where ϵ is the advantage of any PPT breaking the CPA-security of SKE.

Putting everything together, we have $|\Pr_0[\mathsf{Succ}] - \frac{1}{2}| \leq |\Pr_0[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]| + |\Pr_1[\mathsf{Succ}] - \frac{1}{2}| = |\Pr_0[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]| + |\Pr_2[\mathsf{Succ}] - \frac{1}{2}| \leq \Pr_1[\mathsf{NoHide}] + \epsilon$.

6.3 Strong Existential Token Unforgeability

An adversary breaking the strong existential token unforgeability is supposed to output a ciphertext c, two different delegated tokens dec and dec' such that $(\mathsf{TCE.Dec}_{(dk,dec)}(C) = m) \wedge (\mathsf{TCE.Dec}_{(dk,dec')}(C) = m')$ where $(m \neq m') \wedge (m \neq \perp) \wedge (m' \neq \perp)$. The simulator uses the knowledge of dk to get com from c, and can break the binding property of COM by simply outputting (com, dec, dec').

The reason why this output breaks the binding property of COM is as follows. Since $\perp$ is not returned by the decryption algorithm, the first two requirements $\mathsf{Rec}(1^\ell, com, dec) \neq \perp$ and $\mathsf{Rec}(1^\ell, com, dec') \neq \perp$ hold. Let $\tau = \mathsf{Rec}(1^\ell, com, dec)$, $\tau' = \mathsf{Rec}(1^\ell, com, dec')$, and assume to the contrary that $\tau = \tau'$. Note that in the decryption algorithm, $\sigma \leftarrow \mathsf{PKE.Dec}_{dk}(c)$ is independent of whether dec or dec' is used. With the same τ and same σ, $\mathsf{SKE.Dec}_\tau(\sigma)$ must return the same message, contradiction occurs; so $\tau \neq \tau'$, satisfying the last requirement.

6.4 Concrete Instantiation

Any pseudorandom function family (PRF) is sufficient to build a CPA-secure secret key encryption scheme, and commitment scheme can be built solely using collision-free hashing [16]. Both PRF and collision-free hash function are basic cryptographic primitives and are very efficient. For CCA-secure public key encryption in the standard model, one can employ Cramer-Shoup encryption scheme [9], which is secure in the multi-user setting as defined in [3].

In our construction, the delegated token for the decryption is a decommitment string. If we use SHA-256/224 or SHA-512/384 to instantiate the collision-free hash function of the commitment scheme, and takes the decommitment space as a single data block of the hash, the token size is 512 or 1024 bits respectively.

7 Concluding Remarks

This work performs a comprehensive study of token-controlled public key encryption (TCE), a cryptographic primitive that offers many possibilities of application in financial or legal scenario, e.g. rapid distribution of confidential information.

For applications, we contrast the differences between TCE and timed-release encryption (TRE), and discuss many scenarios where TCE is useful. From a theoretical aspect, we generalize and strengthen the existing notion; for example, a partial decryption oracle is available to the adversary. We also address the privacy issues related to a malicious SEM by introducing the notion of release-condition confidentiality and ciphertext unlinkability.

We also show that the only existing scheme with token unforgeability [15] is insecure against insider attack in their definition, and insecure against outsider attack in our new definition. In view of this, we give a secure yet simple and efficient construction in the standard model.

Our generic construction, taking components of different security level, yields TCE with different security properties. Below we give two possible variants. The first achieves strong insider indistinguishability (i.e. embedded-token decryption oracle is provided). Such a scheme can be built from a stateless secret key encryption secure against adaptive chosen ciphertext attack (CCA) [12], with Dodis and Katz's technique for achieving CCA security of multiple encryption [13].

The second one considers "related message attack" [3], such that the adversary has an access of a "left-or-right" encryption oracle [3] for different public keys/tokens (our notion considered encrypting the same message under different public keys), which models the scenario where an attacker can see the ciphertext of related messages. To achieve this we require public key encryption scheme [3] and secret key encryption scheme [4] that are secure in a multi-user setting.

There are still a lot of research problems in this area. For example, there is no known TRE without pairings or random oracles. Although it has been shown in [6] that certificateless public key encryption (CLE) does not imply TRE, their similar settings make it worthwhile to see if any technique in constructing CLE (e.g. [1], [7]) is helpful. On the other hand, improving the security or efficiency of TCE, and finding more novel applications of TCE, are equally interesting.

References

1. Baek, J., Safavi-Naini, R., Susilo, W., et al.: Certificateless Public Key Encryption Without Pairing. In: Zhou, J., et al. (eds.), [21], pp. 134–148.
2. Baek, J., Safavi-Naini, R., Susilo, W.: Token-Controlled Public Key Encryption. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 386–397. Springer, Heidelberg (2005)
3. Bellare, M., Boldyreva, A., Micali, S.: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: Foundations of Computer Science, FOCS '97, pp. 394–403 (1997)

5. Blake, I.F., Chan, A.C-F.: Scalable, Server-Passive, User-Anonymous Timed Release Cryptography. In: Distributed Computing Systems, ICDCS 2005, pp. 504–513. IEEE Computer Society Press, Los Alamitos (2005)
6. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and Non-interactive Timed-Release Encryption. In: Qing, S., Mao, W., Lopez, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
7. Chow, S.S.M., Boyd, C., Nieto, J.M.G.: Security-Mediated Certificateless Cryptography. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 508–524. Springer, Heidelberg (2006)
8. Chow, S.S.M., Hui, L.C.K., Yiu, S.M., Chow, K.P.: Practical Electronic Lotteries with Offline TTP. Computer Communications 29(15), 2830–2840 (2006)
9. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
10. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional Oblivious Transfer and Timed-Release Encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
11. Dent, A.W., Tang, Q.: Revisiting the Security Model for Timed-Release Public-Key Encryption with Pre-Open Capability. In: Information Security, ISC 2007 (to appear, 2007)
12. Desai, A.: New Paradigms for Constructing Symmetric Encryption Schemes Secure against Chosen-Ciphertext Attack. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 394–412. Springer, Heidelberg (2000)
13. Dodis, Y., Katz, J.: Chosen-Ciphertext Security of Multiple Encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)
14. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is Secure under the RSA Assumption. Journal of Cryptology 17(2), 81–104 (2004)
15. Galindo, D., Herranz, J.: A Generic Construction for Token-Controlled Public Key Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 177–190. Springer, Heidelberg (2006)
16. Halevi, S., Micali, S.: Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 201–215. Springer, Heidelberg (1996)
17. Hwang, Y.H., Yum, D.H., Lee, P.J.: Timed-Release Encryption with Pre-open Capability and Its Application to Certified E-mail System. In: Zhou, J., et al. (eds.) [21], pp. 344–358
18. Nali, D., Adams, C.M., Miri, A.: Time-Based Release of Confidential Information in Hierarchical Settings. In: Zhou, J., et al. (eds.) [21], pp. 29–43 (2005)
19. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock Puzzles and Timed-release Crypto. Technical Report MIT/LCS/TR-684 (1996)
20. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332 (2004)
21. Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.): ISC 2005. LNCS, vol. 3650. Springer, Heidelberg (2005)

Trapdoor Permutation Polynomials of $\mathbb{Z}/n\mathbb{Z}$ and Public Key Cryptosystems
(Extended Abstract)

Guilhem Castagnos[1] and Damien Vergnaud[2,⋆]

[1] DMI-XLIM, Université de Limoges,
123, avenue Albert Thomas, 87060 Limoges CEDEX, France
guilhem.castagnos@unilim.fr
[2] École normale supérieure
Département d'informatique, 45 rue d'Ulm, 75230 Paris cedex 05, France
damien.vergnaud@di.ens.fr

Abstract. We define new algorithmic problems and discuss their properties (in particular, we present a careful study of their computational complexity). We apply the new problems to design public key encryption protocols with semantic security relative to their decisional variants. We then show how to provide efficient schemes that are semantically secure under adaptive chosen ciphertext attacks in the random oracle model. Finally, we show that the ideas developed in this extended abstract can be used to design the most efficient known cryptosystem with semantic security under non-adaptive chosen ciphertext attacks in the standard security model.

Keywords: Public Key Encryption, Semantic Security, Standard Model, Random Oracle Model, Chosen-Ciphertext Attacks, Polynomial Diffie-Hellman Problems.

1 Introduction

This paper describes new algorithmic problems using trapdoor permutation polynomials of $\mathbb{Z}/n\mathbb{Z}$ and new constructions of semantically secure public key cryptosystem based on these problems, relative to different scenarios of attacks.

Background. A *trapdoor permutation* is a one-to-one function f that anyone can compute efficiently; however, inverting f is hard unless some "trapdoor" information is also given. Naively, a trapdoor permutation defines a simple public key encryption scheme: the description of f is the public key and the trapdoor is the secret key.

In 1978, Rivest, Shamir, and Adleman proposed the first candidate trapdoor permutation [24]. The RSA setup consists of choosing two distinct large prime

⋆ This work was done while this author was a postdoctoral fellow in the Computer Security group of the Bonn/Aachen International Center for Information Technology.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 333–350, 2007.

numbers p and q, and computing the RSA modulus $n = pq$. The public key is n together with an exponent e (relatively prime to $\varphi(n) = (p-1)(q-1)$). The secret key d is defined to be the multiplicative inverse of e modulo $\varphi(n)$. Encryption and decryption of a message in $(\mathbb{Z}/n\mathbb{Z})^\times$ are defined as follows: $\mathcal{E}(m) = m^e \mod n$ and $\mathcal{D}(c) = c^d \mod n$. Hence, the encryption function is an evaluation of the polynomial X^e of $\mathbb{Z}/n\mathbb{Z}[X]$ and the decryption is performed with the polynomial X^d.

In 1993, in [12], Demytko has suggested to replace the monomial X^e by division polynomials of "elliptic curves" defined over the ring $\mathbb{Z}/n\mathbb{Z}$. The same year, Smith and Lennon [26] have proposed a system, LUC, which uses a special type of Lucas sequences. The encryption and decryption functions can also be seen as an evaluation of a polynomial of $\mathbb{Z}/n\mathbb{Z}[X]$, a Dickson polynomial (cf. [19]). As a consequence, the LUC cryptosystem is very similar to a system already proposed by Müller and Nöbauer (cf. [20,21]).

The security goal for a public key encryption scheme is to guarantee that no partial information about a plaintext message is revealed from its ciphertext, a notion often called *semantic security* or *indistinguishability of ciphertexts* [15]. Unfortunately, the naive public key system built from the three mentioned trapdoor functions is deterministic and hence cannot achieve this security notion. Under a slightly stronger assumption than the intractability of the integer factorization, these primitives give a cryptosystem that is only One-Way under Chosen-Plaintext Attacks (a very weak level of security). The main purpose of the present paper is to propose new combinations of these three polynomial functions giving rise to semantically secure public key cryptosystem.

Several models of attacks have been defined. An encryption scheme that is semantically secure under a Chosen-Plaintext Attack (*resp.* a non-Adaptive Chosen-Ciphertext Attack, *resp.* an Adaptive Chosen-Ciphertext Attack) is said to be IND-CPA secure (*resp.* IND-CCA1 secure, *resp.* IND-CCA2 secure). In [2], it is shown that IND-CCA2 is strictly the strongest notion of semantic security, IND-CCA1, the intermediary notion, and IND-CPA strictly the weakest. Indistinguishably against Chosen-Ciphertext attack is considered to be the correct notion of security for general-purpose public key encryption schemes.

Contributions of the paper. In this paper, we present new algorithmic problems (in section 2, after some notations). Then, in section 3, we give some arguments to validate the cryptographic purpose of those problems, with a careful study of their difficulty and their relations. It is possible to apply the new problems to design public key encryption protocols with semantic security (IND-CPA) relative to the decisional variant of them. This is done in section 4. We then show, in section 5, how these schemes can also be made IND-CCA2 secure assuming the intractability of our decisional RSA variants by using well-known techniques (in the random oracle model [5]). Finally, in section 6, we explain how the ideas developed in this extended abstract can be used to design encryption schemes with higher security in the standard model: for instance, we show that it is possible to construct the most efficient known IND-CCA1 secure cryptosystem with security analysis in the standard model.

2 Permutation Polynomials and New Algorithmic Problems

2.1 Notations

Let $\mathcal{A}$ be a probabilistic Turing machine running in expected polynomial time (a PPT, for short), and let x be an input for $\mathcal{A}$. The probability space that assigns to a string σ the probability that $\mathcal{A}$, on input x, outputs σ is denoted by $A(x)$. Given a probability space S, a PPT that samples a random element according to S is denoted by $x \xleftarrow{R} S$. For a finite set X, $x \xleftarrow{R} X$ denotes a PPT that samples a random element uniformly at random from X. We will use poly and negl to denote respectively unspecified polynomial any negligible[1] functions.

For any integer $k \geq 2$, we denote $\mathsf{Primes}(k)=\{p \in \mathbb{N}, 2^k < p < 2^{k+1}, p \text{ is prime}\}$ and $2\mathsf{Factor}(k) = \{n \in \mathbb{N}, n = pq, \text{ with } p < q < 2p \text{ and } p, q \in \mathsf{Primes}(k)\}$. For $n \in \mathbb{N}$, $\varphi(n) = \#(\mathbb{Z}/n\mathbb{Z})^\times$ denotes the Euler totient value of n.

For two algorithmic problems A and B, we denote $A \xleftarrow{\mathcal{P}} B$ whenever A is polynomial-time reducible to B, and $A \wedge B$ the problem of solving together A and B.

2.2 RSA and LUC

Let $k \geq 2$, $n = pq \in 2\mathsf{Factor}(k)$ and e be an integer relatively prime to $\varphi(n)$. It is well-known that the polynomial X^e of $\mathbb{Z}/n\mathbb{Z}[X]$ induces a permutation of $(\mathbb{Z}/n\mathbb{Z})^\times$. The RSA encryption corresponds to an evaluation of this polynomial. Moreover, this polynomial has a trapdoor: knowing d such that $ed \equiv 1 \mod \varphi(n)$ allows one to invert the evaluation of this polynomial at any point.

Another permutation polynomial is the LUC function used in the system of [26]. Given two integers a and b such that $a^2 - 4b$ is a non-square, the Lucas sequence V is given by a second-order linear recurrence relation:

$$\forall k \geqslant 1,\ V_{k+1}(a,b) = aV_k(a,b) - bV_{k-1}(a,b),\ V_1(a,b) = a,\ V_0(a,b) = 2.$$

Let e be an integer relatively prime to $(p^2-1)(q^2-1)$. The LUC function, $x \mapsto V_e(x,1) \pmod n$, is a permutation of the set $\{x \in \mathbb{N},\ 0 < x < n,\ \gcd(x,n) = 1,\ \gcd(x^2-4,n) = 1\}$, whose inverse is $x \mapsto V_d(x,1) \pmod n$, where d is the multiplicative inverse of e modulo $(p^2-1)(q^2-1)$ (see [6] for more details on the LUC function). One can see that $V_e(X,1)$ is in fact a polynomial of degree e,

$$V_e(X,1) = \sum_{i=0}^{\lfloor e/2 \rfloor} \frac{e}{e-i} \binom{e-i}{i} X^{e-2i},$$

which is a special type of Dickson polynomial (cf. [19]) and, for that reason, a permutation polynomial.

[1] *I. e.*, $\forall c \geq 0, \exists K_c \in \mathbb{N}, \forall k \in [\![K_c, +\infty[\![, \mathsf{negl}(k) \leq k^{-c}$.

These two polynomials of degree e derived from the RSA and LUC cryptosystems can be evaluated at low cost. If we denote $|e|$ the size of e in bits, the evaluation of the RSA polynomial needs $(3/2)|e|$ multiplications modulo n on the average, with the square and multiply algorithm, and $2|e|$ multiplications are needed for the evaluation of the LUC polynomial, using the algorithm of [17].

In the following, we combine these two polynomials to define new algorithmic problems and build new systems. We will define our new problems in a general setting, by considering arbitrary permutation polynomials of $(\mathbb{Z}/n\mathbb{Z})^\times$. The study will be identical as if we were working directly with the RSA and LUC polynomials except for one thing: as the RSA polynomial induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$ some extra reductions will be possible. Consequently, this specific case of polynomial Q such that $Q(xy) = Q(x)Q(y)$ for all elements x, y of $(\mathbb{Z}/n\mathbb{Z})^\times$ will be considered in the study of our problems.

Remark 1. In order to design cryptosystems, one can also follow the ideas of Schwenk and Huber (cf. [25]), and use more general permutation polynomials for which the inverse function is not explicit.

2.3 Permutation Polynomials and Pointwise Inversion

In this paragraph, we define the problem of pointwise inversion of an arbitrary permutation polynomial (PP).

Definition 1. *A* PP generator *is a PPTM that takes a security parameter k as input and outputs a 4-tuple (n, p, q, P) where $n = pq \in 2\mathsf{Factor}(k)$ and $P \in \mathbb{Z}/n\mathbb{Z}[X]$ is a permutation of $(\mathbb{Z}/n\mathbb{Z})^\times$ which can be evaluated at any value of $(\mathbb{Z}/n\mathbb{Z})^\times$ in polynomial time in k. Let $e : \mathbb{N} \longrightarrow \mathbb{N}$. A PP generator Gen is said to be a PP generator of* degree e *if for any $k \in \mathbb{N}$ and any $(n, p, q, P) \leftarrow \mathsf{Gen}(k)$, $\deg(P) \leq e(k)$.*

The next definition quantifies the resistance to pointwise inversion for PP generators (*i. e.*, the problem: given $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$, compute $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ denoted $P^{-1}(n)$).

Definition 2. *Let Gen be a PP generator. Let $\mathcal{A}$ be a PPTM that takes as input a triple $(n, P, y) \in \mathbb{N} \times \mathbb{Z}/n\mathbb{Z}[X] \times (\mathbb{Z}/n\mathbb{Z})^\times$ and outputs an element $x \in (\mathbb{Z}/n\mathbb{Z})^\times$. We consider the following random experiments, where k is a security parameter:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{P}^{-1}}_{\mathsf{Gen},\mathcal{A}}(k)}$$
$$(n, p, q, P) \xleftarrow{R} \mathsf{Gen}(k)$$
$$y \xleftarrow{R} (\mathbb{Z}/n\mathbb{Z})^\times$$
$$x \leftarrow \mathcal{A}(n, P, y)$$
$$\textit{Return } 1 \textit{ if } P(x) = y, \ 0 \textit{ otherwise}$$

The success *of $\mathcal{A}$ in solving the pointwise inversion problem is*

$$\mathbf{Succ}^{\mathsf{P}^{-1}}_{\mathsf{Gen},\mathcal{A}}(k) = \Pr[\mathbf{Exp}^{\mathsf{P}^{-1}}_{\mathsf{Gen},\mathcal{A}}(k) = 1].$$

Let τ be an integer and ε a real in $[0,1]$. Gen *is said to be (k,τ,ε)-P^{-1}-secure if no adversary $\mathcal{A}$ running in time τ has success* $\mathbf{Succ}^{\mathsf{P}^{-1}}_{\mathsf{Gen},\mathcal{A}}(k) \geq \varepsilon$.

2.4 Permutation Polynomials and Polynomial Diffie-Hellman Problem

Definition 3. *A* PDH generator *is a PPTM that takes a security parameter k as input and outputs a 5-tuple (n,p,q,P,Q,R) where $n = pq \in 2\mathsf{Factor}(k)$, $P \in \mathbb{Z}/n\mathbb{Z}[X]$ and $Q \in \mathbb{Z}/n\mathbb{Z}[X]$ are permutations of $(\mathbb{Z}/n\mathbb{Z})^\times$ which can be evaluated at any value of $(\mathbb{Z}/n\mathbb{Z})^\times$ in polynomial time in k and $R \in \mathbb{Z}/n\mathbb{Z}[X,Y]$ is a bivariate polynomial which can be evaluated at any value of $(\mathbb{Z}/n\mathbb{Z})^{\times 2}$ in polynomial time in k. Let $e_P : \mathbb{N} \longrightarrow \mathbb{N}$, $e_Q : \mathbb{N} \longrightarrow \mathbb{N}$ and $e_R : \mathbb{N} \longrightarrow \mathbb{N}$. A PDH generator* Gen *is said to be a PDH generator of* degree (e_P, e_Q, e_R) *if for any $k \in \mathbb{N}$ and any $(n,p,q,P,Q,R) \leftarrow \mathsf{Gen}(k)$, $\deg(P) \leq e_P(k)$, $\deg(Q) \leq e_Q(k)$, $\deg_X(R) \leq e_R(k)$.*

We now define a new family of algorithmic problems: the *computational polynomial Diffie-Hellman problems* that generalizes the pointwise inversion problem:

Computational Polynomial DH: C-POL-DH(n,P,Q,R)
Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$;
Find: $R(a,b) \in (\mathbb{Z}/n\mathbb{Z})^\times$.

The problem is named after the Diffie-Hellman key exchange [13] because of its similarity with it in the special case where $Q = P$ and $R(X,Y) = P(XY)$. In this paper, we deal only with the cases $R(X,Y) = XY$, $R(X,Y) = P\big((XY)^\ell\big)$ and $R(X,Y) = Q(X)$ that we denote respectively C-POL1(n,P,Q), C-POL2(n,ℓ,P,Q) and C-DPOL(n,P,Q).

The next definition quantifies the resistance to the computational polynomial Diffie-Hellman problems for PDH generators.

Definition 4. *Let* Gen *be a PDH generator. Let $\mathcal{A}$ be a PPTM that takes as input a 6-tuple $(n,P,Q,R,y,z) \in \mathbb{N} \times \mathbb{Z}/n\mathbb{Z}[X]^2 \times \mathbb{Z}/n\mathbb{Z}[X,Y] \times (\mathbb{Z}/n\mathbb{Z})^{\times 2}$ and outputs an element $z \in (\mathbb{Z}/n\mathbb{Z})^\times$. We consider the following random experiments, where k is a security parameter:*

Experiment $\mathbf{Exp}^{\text{C-POL-DH}}_{\mathsf{Gen},\mathcal{A}}(k)$

$(n,p,q,P,Q,R) \xleftarrow{R} \mathsf{Gen}(k)$
$y \xleftarrow{R} (\mathbb{Z}/n\mathbb{Z})^\times$, $y' \xleftarrow{R} P(y)$
$z \xleftarrow{R} (\mathbb{Z}/n\mathbb{Z})^\times$, $z' \xleftarrow{R} Q(z)$
$x \leftarrow \mathcal{A}(n,P,Q,R,y',z')$
Return 1 *if* $x = R(y,z)$, 0 *otherwise*

The success *of $\mathcal{A}$ in solving the computational polynomial DH problem is*

$$\mathbf{Succ}^{\text{C-POL-DH}}_{\mathsf{Gen},\mathcal{A}}(k) = \Pr[\mathbf{Exp}^{\text{C-POL-DH}}_{\mathsf{Gen},\mathcal{A}}(k) = 1].$$

Let τ be an integer and $\varepsilon \in [0,1]$. Gen *is said to be (k,τ,ε)-*C-POL-DH*-secure if no adversary $\mathcal{A}$ running in time τ has success* $\mathbf{Succ}^{\text{C-POL-DH}}_{\mathsf{Gen},\mathcal{A}}(k) \geq \varepsilon$.

Now, we define the decision problem D-POL-DH(n, P, Q, R) where an element from $(\mathbb{Z}/n\mathbb{Z})^\times$ is given and the algorithm has to decide whether it is a valid candidate for the C-POL-DH(n, P, Q, R) problem.

Decision Polynomial DH: D-POL-DH(n, P, Q, R)
Given: $\alpha = P(a) \in (\mathbb{Z}/n\mathbb{Z})^\times$, $\beta = Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$ and $\gamma \in (\mathbb{Z}/n\mathbb{Z})^\times$;
Decide whether: $\gamma = R(a, b)$.

We define three decision problems D-POL1(n, P, Q), D-POL2(n, ℓ, P, Q) and finally D-DPOL(n, P, Q) for the cases $R(X, Y) = XY$, $R(X, Y) = P\big((XY)^\ell\big)$ and $R(X, Y) = Q(X)$ (respectively).

Remark 2. The D-DPOL(n, P, Q) decision problem can be rewritten (the same holds for the computational problem) as follows: given $P(a)$ and $\gamma \in (\mathbb{Z}/n\mathbb{Z})^\times$ decide whether $\gamma = Q(a)$. Hence, the C-DPOL(n, P, Q) and D-DPOL(n, P, Q) problems are generalisations of the Dependent-RSA problems defined in [23].

3 Relations Among the New Problems

In this section, we discuss the problems defined in the previous section with a careful study of both their difficulty and their relations. For clarity reasons, when the reductions are simple, the theorems are stated with less formalism than the definitions of the previous section. Throughout this section, for a security parameter k, n, P and Q will correspond to the output of a *PDH generator* on input k. For short, we will denote e_p and e_q the degrees of P and Q.

We define an extraction problem, E-POL-DH(n, P, Q, R): Given $P(a)$, $Q(b)$ and $R(a, b)$, find a and b. We denote as before E-POL1, E-POL2, E-DPOL the extraction problems for the special values of R.

We first study the C-POL1 and C-POL2 classes of problems, then we will analyse the C-DPOL class of problem and finally the relation between these three classes.

3.1 The C-POL1 and C-POL2 Problems

For the C-POL1 problem, we have the straightforward theorem:

Theorem 1. D-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longleftarrow}$ C-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1}(n) \wedge Q^{-1}(n)$.

Proof. All the reductions follow from the definition of the C-POL1(n, P, Q) problem except C-POL1$(n, P, Q) \overset{\mathcal{P}}{\Longrightarrow} P^{-1}(n) \wedge Q^{-1}(n)$. Suppose that we know $P(a)$ and we want to compute a. We choose a random b in $(\mathbb{Z}/n\mathbb{Z})^\times$ and we give the value $P(a)$ and $Q(b)$ to an oracle for C-POL1(n, P, Q) which gives the value ab in reply, so we can recover b. We can invert Q with a symmetric process. □

For the C-POL2 problem, we use the extraction problem to state a similar theorem.

Theorem 2. *For an RSA integer n, and two permutation polynomials P and Q of $(\mathbb{Z}/n\mathbb{Z})^\times$,*

$$\text{C-POL2} \wedge \text{E-POL2} \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1} \wedge Q^{-1} \overset{\mathcal{P}}{\Longrightarrow} \begin{matrix}\text{C-POL2}\\ \text{E-POL2}\end{matrix} \overset{\mathcal{P}}{\Longrightarrow} \text{D-POL2}\,. \qquad \square$$

We now examine the difficulty of the decision problems.

Difficulty of D-POL1 and D-POL2. The best known way to solve these problems is to solve the corresponding extraction problem (cf. [8]). We know the values of $P(a)$, $Q(b)$ and $R(a,b)$ and we want to find the values of a and b. Suppose that $e_Q \leqslant e_P$ (else the attack is done with the symmetric method). We compute the resultant with respect to the variable Y:

$$S(X) = \operatorname*{Res}_Y(R(X,Y) - R(a,b),\ Q(Y) - Q(b)).$$

This gives a polynomial $S(X)$ of degree $e_R e_Q$ with $S(a) = 0$, so

$$(X - a) \mid \gcd(S(X), P(X) - P(a)).$$

In fact, in many cases[2], we will have $(X - a) = \gcd(S(X), P(X) - P(a))$, and this method allows to recover a. If we are trying to solve the E-POL1 problem we know ab, so we have also recovered b. Else, for the E-POL2 problem, we can recover b by computing $\gcd(R(a,Y) - R(a,b), Q(Y) - Q(b))$.

Remark 3. In the previous description, "the" resultant and "the" gcd of two polynomials with coefficients in $\mathbb{Z}/n\mathbb{Z}$ are understood as the results of the classical algorithms (described in [14] for instance) which compute the resultant and the gcd of polynomials over a field. In the unlikely event, that a non-trivial factor of n appears during this computation, the adversary simply aborts the computation and uses this knowledge to solve the instance of its problem.

The resultant can be computed with $\mathcal{O}(e_R^2 e_Q \log^2(e_R e_Q) \log\log(e_R e_Q))$ operations in $\mathbb{Z}/n\mathbb{Z}$, according to [14, Corollary 11.18, p. 310]. Note that $e_R = 1$ for E-POL1 and $e_R = \ell e_P$ for E-POL2, so, if ℓ is large enough, this method will be infeasible even if e_P is small.

According to [14, Corollary 11.6, p. 304], the computation of the first gcd can be done in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R e_Q, e_P)$ and the computation of the second gcd in $\mathcal{O}(e \log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_R, e_Q)$. This complexity of attacks on the extraction problems will be used in the next section to set key sizes for the cryptosystems that we will built.

Now, suppose that the polynomial Q induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$ (in particular, if Q is associated to the RSA function). In this case, it is possible to make another reduction from $Q^{-1}(n)$ to C-POL2(n, ℓ, P, Q) when $\ell = 1$.

[2] Some experimentations confirm that, but, we have not been able to prove this fact. Note that in [8,23] a similar fact is stated, without proof. Anyway, as our goal is to estimate the size of the degrees to make our cryptosystems secure, only the possibility of an attack matters.

Theorem 3. *Let* $e_P : \mathbb{N} \longrightarrow \mathbb{N}$, $e_Q : \mathbb{N} \longrightarrow \mathbb{N}$ *and* $e_R : \mathbb{N} \longrightarrow \mathbb{N}$ *and let* Gen *be a PDH generator of degree* (e_P, e_Q, e_R). *Suppose that, for any* $k \in \mathbb{N}$ *and any* $(n, p, q, P, Q, R) \in \mathsf{Gen}(k)$, Q *is a morphism of* $(\mathbb{Z}/n\mathbb{Z})^\times$ *and* P *is not a polynomial in* X^i *for any* $i > 1$. *Let* $\tau \in \mathbb{N}^{\mathbb{N}}$, $\varepsilon \in [0,1]^{\mathbb{N}}$ *and* $\mathcal{A}$ *be an adversary that* $(k, \tau(k), \varepsilon(k))$*-solves the* C-POL2$(n, 1, P, Q)$ *problem for any integer* $k \in \mathbb{N}$. *There exists an algorithm* $\mathcal{B}$ *that* $(k, \tau'(k), \varepsilon'(k))$*-solves the* $Q^{-1}(n)$ *problem such that*

$$\varepsilon' \geq \varepsilon^{e_P} - \mathsf{negl} \text{ and } \tau' \leq e_P \cdot \tau + e_P^3 \cdot \mathsf{poly}.$$

Proof. The algorithm $\mathcal{A}$ takes as input an instance of the C-POL2$(n, 1, P, Q)$ problem: for any $k \in \mathbb{N}$, given $(n, p, q, P, Q, R) \in \mathsf{Gen}(k)$, $P(a)$ and $Q(b)$ in $(\mathbb{Z}/n\mathbb{Z})^\times$, he will return $P(ab) \in (\mathbb{Z}/n\mathbb{Z})^\times$ in time at most $\tau(k)$ with probability at least $\varepsilon(k)$.

Let $\mathcal{I}$ be the subset of $\{1, \ldots, e_P(k)\}$ of cardinality $m > 1$ such that $P(X) = \sum_{i \in \mathcal{I}} p_i X^i$, where all the $(p_i)_{i \in \mathcal{I}}$ are non-zero elements of $(\mathbb{Z}/n\mathbb{Z})^\times$. Since P is not a polynomial in X^i for any $i > 1$, the gcd of $\mathcal{I}$ is equal to 1.

Given an element $Q(b) \in (\mathbb{Z}/n\mathbb{Z})^\times$, the algorithm $\mathcal{B}$ will recover b. It starts by choosing randomly m couples $(s_j, t_j) \in (\mathbb{Z}/n\mathbb{Z})^\times \times (\mathbb{Z}/n\mathbb{Z})^\times$, with $j \in \{1, \ldots, m\}$ so that all the s_j and the t_j with $j \in \{1, \ldots, m\}$ are distinct.

For each $j \in \{1, \ldots, m\}$, $\mathcal{B}$ gives the values $P(s_j)$ and $Q(bt_j) = Q(b)Q(t_j)$ to the algorithm $\mathcal{A}$ which returns the value of $P(s_j t_j b)$ with probability at least $\varepsilon(k)^m$ (since the m queries are independent). After these queries, $\mathcal{B}$ gets the m equations:

$$\sum_{i \in \mathcal{I}} p_i (s_j t_j)^i b^i = P(s_j t_j b), \text{ for } j \in \{1, \ldots, m\}$$

with the m unknowns $(b^i)_{i \in \mathcal{I}}$. If we denote $\mathcal{I} := \{i_1, i_2, \ldots, i_m\}$, with $0 < i_1 < i_2 < \cdots < i_m = e_P$, the system of equations is associated with the following matrix:

$$M := \begin{bmatrix} p_{i_1}(s_1 t_1)^{i_1} & p_{i_2}(s_1 t_1)^{i_2} & \cdots & p_{i_m}(s_1 t_1)^{i_m} \\ \vdots & \vdots & & \vdots \\ p_{i_1}(s_m t_m)^{i_1} & p_{i_2}(s_m t_m)^{i_2} & \cdots & p_{i_m}(s_m t_m)^{i_m} \end{bmatrix}$$

The method successes if $\det(M) \in (\mathbb{Z}/n\mathbb{Z})^\times$. We focus on the study of $\det(M) \neq 0$ (another value of $(\mathbb{Z}/n\mathbb{Z}) \setminus (\mathbb{Z}/n\mathbb{Z})^\times$ will reveal the factorisation of n).

We have

$$\det(M) = \left(\prod_{j=1}^{m} p_{i_j} c_j^{i_1} \right) \begin{vmatrix} 1 & c_1^{i_2 - i_1} & \cdots & c_1^{i_m - i_1} \\ \vdots & \vdots & & \vdots \\ 1 & c_m^{i_2 - i_1} & \cdots & c_m^{i_m - i_1} \end{vmatrix}$$

where $c_j := s_j t_j$ for $j = 1, \ldots, m$. This last determinant, D, is a generalized Vandermonde determinant. One can see that

$$D = \left(\prod_{1 \leq i < j \leq m} (c_j - c_i) \right) T(c_1, c_2, \ldots, c_m),$$

where T is a polynomial of degree $i_m - i_1 - m + 1$ in c_m (see [11], for example, for details on this polynomial). So, if all the $(c_j)_{j=1,\ldots,m}$ are distinct, once all the $(s_j)_{j=1,\ldots,m}$, all the $(t_j)_{j=1,\ldots,m-1}$ have been chosen, less than $(i_m - i_1 - m + 1)^2$ values of t_m can make the method fail.

So with standard Gauss elimination, $\mathcal{B}$ can recover the $(b^i)_{i\in\mathcal{I}}$ with $\mathcal{O}(e_P(k)^3)$ operations in $\mathbb{Z}/n\mathbb{Z}$ and m independent queries to the oracle. As $\gcd(\mathcal{I}) = 1$, there exists a linear combination of the elements of $\mathcal{I}$ that equals 1, therefore $\mathcal{B}$ can recover b. □

Remark 4. If $\ell > 1$, with the method used in the proof, we can only recover the value of b^ℓ. Then, we can recover b by computing $\gcd(Q(Y) - Q(b), Y^\ell - b^\ell)$.

3.2 The C-DPOL Problem

As shown in Remark 2, the C-DPOL can be rewritten: Given $P(a)$, find $Q(a)$; and the extraction problem, E-DPOL, can be rewritten: Given $P(a)$ and $Q(a)$, find a. We then have the following (straightforward) theorem, that generalizes ([23], Theorem 3):

Theorem 4. *Let* Gen *be a PDH generator. We have:*

$$\text{C-DPOL} \wedge \text{E-DPOL} \overset{\mathcal{P}}{\Longleftrightarrow} P^{-1} \overset{\mathcal{P}}{\Longrightarrow} \begin{matrix}\text{C-DPOL}\\ \text{E-DPOL}\end{matrix} \overset{\mathcal{P}}{\Longrightarrow} \text{D-DPOL}\,.$$

Now let's try to solve the E-DPOL problem. We know the values of $P(a)$ and $Q(a)$ and we want to compute the value of a. We have $(X-a)$ divides $\gcd(P(X)-P(a), Q(X)-Q(a))$ and again, in many cases, we will have an equality. The complexity of the computation of the gcd is $\mathcal{O}(e\log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$, where $e = \max(e_Q, e_P)$. If e_Q and e_P are greater than, say 2^{60}, this method will fail. This method for solving E-DPOL problem allows to break the D-DPOL problem. In conjunction with Theorem 4, this method also leads to a reduction from the $P^{-1}(n)$ problem to the C-DPOL(n, P, Q) problem in $\mathcal{O}(e\log^2 e \log\log e)$ operations in $\mathbb{Z}/n\mathbb{Z}$.

Again, suppose that the polynomial P induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$: we can also make another reduction from C-DPOL(n, P, Q) to P^{-1}.

Theorem 5. *Let* $e_P : \mathbb{N} \longrightarrow \mathbb{N}$, $e_Q : \mathbb{N} \longrightarrow \mathbb{N}$ *and* $e_R : \mathbb{N} \longrightarrow \mathbb{N}$ *and let* Gen *be a PDH generator of degree* (e_P, e_Q, e_R). *Suppose that, for any* $k \in \mathbb{N}$ *and any* $(n, p, q, P, Q, R) \in \mathsf{Gen}(k)$, P *is a morphism of* $(\mathbb{Z}/n\mathbb{Z})^\times$ *and* Q *is not a polynomial in* X^i *for any* $i > 1$. *Let* $\tau \in \mathbb{N}^{\mathbb{N}}$, $\varepsilon \in [0,1]^{\mathbb{N}}$ *and* $\mathcal{A}$ *be an adversary that* $(k, \tau(k), \varepsilon(k))$*-solves the* C-DPOL$(n, P, Q)$ *problem for any integer* $k \in \mathbb{N}$. *There exists an algorithm* $\mathcal{B}$ *that* $(k, \tau'(k), \varepsilon'(k))$*-solves the* $P^{-1}(n)$ *problem such that*

$$\varepsilon' \geq \varepsilon^{e_Q} - \mathsf{negl} \text{ and } \tau' \leq e_Q \cdot \tau + e_Q^3 \cdot \mathsf{poly}.$$

□

The proof is analogous to that of Theorem 3.

3.3 Relations Among the Three Classes of Problems

It is trivial to see that for all $\ell \geq 1$,

$$\begin{array}{ccc} \text{C-POL1}(n,P,Q) & \overset{\mathcal{P}}{\Longrightarrow} & \text{D-POL1}(n,P,Q) \\ \Downarrow \mathcal{P} & & \Uparrow \mathcal{P} \\ \text{C-POL2}(n,\ell,P,Q) & \overset{\mathcal{P}}{\Longrightarrow} & \text{D-POL2}(n,\ell,P,Q) \end{array}$$

In the special case where the polynomial P induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$, as $P((ab)^\ell) = P(ab)^\ell$ and $P(ab) = P(a)P(b)$, we have the following theorem.

Theorem 6. *If P induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$,*

$$\text{C-POL2}(n,\ell,P,Q) \overset{\mathcal{P}}{\Longleftarrow} \text{C-POL2}(n,1,P,Q) \overset{\mathcal{P}}{\Longleftrightarrow} \text{C-DPOL}(n,Q,P),$$

$$\text{D-POL2}(n,\ell,P,Q) \overset{\mathcal{P}}{\Longrightarrow} \text{D-POL2}(n,1,P,Q) \overset{\mathcal{P}}{\Longleftrightarrow} \text{D-DPOL}(n,Q,P). \qquad \square$$

In fact the idea of the proof of the previous theorem can be used to make a reduction from D-POL1(n,P,Q) to D-DPOL(n,P,Q) if Q is a morphism. Suppose that we know the values of $P(a)$ and $Q(b)$ and that we want to decide if an element c equals ab. If this is the case, $Q(c) = Q(a)Q(b)$. So we can submit $P(a)$ and $Q(c)/Q(b)$ to an oracle of the D-DPOL(n,P,Q) problem to solve the D-POL1(n,P,Q) problem.

As D-DPOL$(n,Q,P) \overset{\mathcal{P}}{\Longleftrightarrow}$ D-DPOL(n,P,Q), we have proved the following theorem:

Theorem 7. *If P or Q induces a morphism of $(\mathbb{Z}/n\mathbb{Z})^\times$,*

$$\text{D-POL1}(n,P,Q) \overset{\mathcal{P}}{\Longleftarrow} \text{D-DPOL}(n,P,Q). \qquad \square$$

4 IND-CPA-Secure Public Key Cryptosystems

Let f be a trapdoor permutation and g be another function with the following pseudo-randomness property: the distribution of $(f(k), g(k))$ induced by a random k cannot be distinguished (by a polynomially bounded adversary) from a randomly distributed $(f(k), r)$. Then the encryption $E(m) = (f(k), g(k) \oplus m)$ is semantically secure (cf. [23,7]). In this section, we revisit this approach by using for the function g a trapdoor permutation.

Following this paradigm, we define three new encryption schemes where the public key is (n,P,Q) or (n,P,Q,R) and the corresponding secret key is P^{-1} or (P^{-1},Q^{-1}), with the notations of the previous section (*i. e.*, n, P, Q, R correspond to the output of a *PDH generator* for a given security parameter).

To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto \big(P(r_0), Q(r_1), mR(r_0, r_1)\big)$
Function 2: $(m, r) \mapsto \big(P(r), mQ(r)\big)$
Function 3: $(m, r) \mapsto \big(P(mr), Q(r^{-1})\big)$

To decrypt, a user uses his knowledge P^{-1} or (P^{-1}, Q^{-1}) to recover r or (r_0, r_1) then m.

Theorem 8. *The previous schemes are One-Way and semantically secure under Chosen Plaintext Attack relative to the following problems:*

Encryption function	*One-Wayness*	*Semantic security*
Function *1,* $R(X, Y) = XY$	C-POL1(n, P, Q)	D-POL1(n, P, Q)
Function *1,* $R(X, Y) = P\big((XY)^\ell\big)$	C-POL2(n, ℓ, P, Q)	D-POL2(n, ℓ, P, Q)
Function *2*	C-DPOL(n, P, Q)	D-DPOL(n, P, Q)
Function *3*	C-POL1(n, P, Q)	D-POL1$(n, P, Q)^{(*)}$

$^{(*)}$ *If P or Q is a morphism.*

Proof (Sketch). For the first three schemes, the proof relies on the analysis done in [7]. The fourth encryption scheme mixes the one-time-pad masking approach used above with the trapdoor property of the function induced by P. The semantic security of this scheme can be rewritten: there is no polynomial algorithm that can choose a value $m \in (\mathbb{Z}/n\mathbb{Z})^\times$ and then recognize the couples $(P(a), Q(b)) \in (\mathbb{Z}/n\mathbb{Z})^\times \times (\mathbb{Z}/n\mathbb{Z})^\times$, satisfying $ab = m$. It's easy to see that if P or Q is a morphism, this assertion is equivalent to the intractability of the D-POL1(n, P, Q) decision problem. □

Efficiency considerations. From the encryption functions above, we design five practical cryptosystems, three with Function 1, by setting $R(X, Y) = XY$, $R(X, Y) = P(XY)$ and $R(X, Y) = P\big((XY)^\ell$ with $\ell > 1$; one with Function 2; and one with Function 3.

For the polynomial P we use the LUC polynomial $V_e(X, 1)$ and for the polynomial Q, the RSA polynomial of the same degree, *i. e.*, $Q(X) = X^e$. In order to compare the efficiency of these schemes, we use an RSA modulus of 1024 bits and we adjust the parameter e (and ℓ) in order to achieve a 2^{80} security. For this, we use Theorem 8 and the analysis done in Section 3. These new cryptosystems and the corresponding values of the parameters are given in the following table.

Scheme	Ciphertext	Public keys
Scheme 1	$V_e(r_o, 1), {r_1}^e, mr_0r_1$	$e = 2^{67} + 3$.
Scheme 2	$V_e(r_o, 1), {r_1}^e, mV_e(r_0r_1)$	$e = 2^{23} + 9$.
Scheme 3	$V_e(r_o, 1), {r_1}^e, mV_e\big((r_0r_1)^\ell\big)$	$e = 5$ and $\ell = 2^{31} + 65$
Scheme 4	$V_e(r, 1), mr^e$	$e = 2^{67} + 3$.
Scheme 5	$V_e(mr, 1), r^{-e}$	$e = 2^{67} + 3$.

Now, we compare the concrete efficiency of our new schemes with the one from [7,23]. For the D-RSA scheme of [23] we use $e = 2^{67} + 3$ and for the scheme of Catalano *et al.* ([7]), we use $e = 2^{16} + 1$. The unity of complexity is the cost of a multiplication modulo n. We use the following estimations: a multiplication modulo n^2 costs as much as three multiplications modulo n, an inversion costs 10 multiplications, a multiplication modulo p costs 1/3 multiplication modulo n and a multiplication modulo p^2 costs one multiplication modulo n. We use the Chinese Remainder Theorem for the decryption process of all schemes. The comparison is done in the following table.

Scheme	D-RSA	Catalano	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5
Input	1024						
Output	2048		3072			2048	
Encryption	139	52	205	119	**44**	204	214
Decryption	567	570	1204	1234	1228	736	1196

One can remark that the new cryptosystems appear to be quite practical. If the decryption phase of these schemes (except Scheme 4) suffers from the cost of the simultaneous inversions of the LUC and RSA function, the encryption process is very fast and Scheme 3 (which is an improvement of Scheme 2) can encrypt *faster* than the D-RSA and Catalano *et al.* cryptosystems. Schemes 1 and 5 have a similar complexity and are the most efficient semantically secure cryptosystems with One-Wayness proved equivalent to the problem of inverting *simultaneously* RSA *and* LUC.

5 IND-CCA2-Secure Public Key Cryptosystems in the ROM

In this section, we apply standard techniques to obtain chosen ciphertext security (from these new primitives) in the random oracle model formalized by Bellare and Rogaway in 1993 [5], in which cryptographic protocols are designed

and proved secure under the additional assumption that publicly available functions that are chosen truly at random exist. These random oracles can only be accessed in a black-box way, by providing an input and obtaining the corresponding output. A similar method is used in [23] for instance.

Let h be a cryptographic hash function (seen like a random oracle). With the previous notations, the public key is now (n, P, Q, h) or (n, P, Q, R, h), and the corresponding secret key is P^{-1} or (P^{-1}, Q^{-1}). To encrypt a message m of $(\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ (or $(r_0, r_1) \in (\mathbb{Z}/n\mathbb{Z})^{\times 2}$) and uses one of the three following encryption functions:

Function 1: $(m, r_0, r_1) \mapsto (P(r_0), Q(r_1), mR(r_0, r_1), h(m||r_0||r_1))$
Function 2: $(m, r) \mapsto (P(r), mQ(r), h(m||r))$
Function 3: $(m, r) \mapsto (P(mr), Q(r^{-1}), h(m||r))$

The decryption process is done as in Section 4 except that the message is returned only if the hash value is correct.

Theorem 9. *The previous schemes are semantically secure against Adaptive Chosen Ciphertext Attack in the Random Oracle Model relative to the following problems:*

Encryption function	*Semantic security*
Function *1*, $R(X, Y) = XY$	D-POL1(n, P, Q)
Function *1*, $R(X, Y) = P\big((XY)^\ell\big)$	D-POL2(n, ℓ, P, Q)
Function *2*	D-DPOL(n, P, Q)
Function *3*	D-POL1$(n, P, Q)^{(*)}$

$^{(*)}$ *If P or Q is a morphism.*

Proof (Sketch). The proof is standard. The random oracle model is simulated in the standard way and the instance of the decision problem is embedded in the challenge ciphertext without updating the hash table. The hash table is used to simulate the decryption oracle: when the adversary makes a decryption query, the reduction looks in this table to get the pair (m, r) or the triple (m, r_0, r_1) corresponding to the last element of the ciphertext. It returns the message m only if the encryption of m with the value r or the pair (r_0, r_1) produces the same ciphertext; otherwise, it returns the reject symbol.

The simulation of the random oracle is perfect and the probability that a decryption query is incorrect is exponentially small in the size of the hash values. Details can be found in [23], for instance. □

Remark 5. As done in [23], it is also possible to modify our schemes in order to make them IND-CCA2 in the random oracle model relative to the corresponding computational problems. This transformation permits to use smaller degree polynomials P and Q in the encryption procedure but unfortunately the resulting schemes are IND-CPA-secure only in the Random Oracle Model.

6 IND-CCA1-Secure Public Key Cryptosystems in the Standard Model

6.1 Damgård's Elgamal

Let $\mathbb{G}$ be an additive group of prime order q, let k be the bit size of the elements of $\mathbb{G}$ and let P be a generator of $\mathbb{G}$. In 1991, Damgård [10] presented a simple variant of the Elgamal encryption scheme in $\mathbb{G}$. In his proposal, Alice publishes two public keys $A_1 = [a_1] \cdot P$ and $A_2 = [a_2] \cdot P$ and keeps secret their discrete logarithms a_1 and a_2. When Bob wants to send privately a message $m \in \{0,1\}^k$ to Alice, he picks uniformly at random an integer $r \in [\![1, q-1]\!]$ and transmits the triple (Q_1, Q_2, C) where $Q_1 = [r] \cdot P$, $Q_2 = [r] \cdot A_1$ and $C = m \oplus ([r] \cdot A_2)$. When she receives the ciphertext (Q_1, Q_2, C), Alice checks whether the equality $Q_2 = [a_1] \cdot Q_1$ holds: if it is the case, she retrieves the message m, as $m = C \oplus ([a_2] \cdot Q_1)$, otherwise she rejects the ciphertext.

Damgård proved that if the DDH problem is hard in $\mathbb{G}$, then this scheme is IND-CCA1-secure, if we assume the so-called *knowledge-of-exponent assumption* [22]. Intuitively this assumption states that, without the knowledge of a_1, the only way to generate couples $(Q_1, Q_2) \in \mathbb{G}^2$, satisfying $Q_2 = [a_1] \cdot Q_1$, is to choose an integer $r \in [\![1, q-1]\!]$ and to compute $Q_1 = [r] \cdot P$ and $Q_2 = [r] \cdot A_1$.

The knowledge-of-exponent assumption is a strong and non-standard one, but to date it has not been proven false. It has been criticized for assuming one can perform "reverse engineering" of an adversary. It should therefore be considered with caution, all the more since Bellare and Palacio [3] showed that a somehow similar assumption used in [16] is false.

The function that maps r to $([r] \cdot P, [r] \cdot A_1)$ is what Damgård called a *One-Way function with sparse image* (*i. e.*, only a very small fraction of $\mathbb{G}^2$ is in its image and it seems computationally infeasible to sample an element of this set without the knowledge of its preimage). Damgård predicts that such One-Way functions would be extremely useful in other contexts. His proposal has indeed found applications in identification and zero-knowledge protocols ([3,4,1,16]), but it has proved to be extremely difficult to find other examples that can be reduced to reasonable assumptions. The purpose of the next paragraph is to explain how our approach can be extended in order to propose such a function.

6.2 Knowledge of Preimage Assumption

Let Gen be a PDH generator. Suppose we are given (n, P_1, P_2, R, y, z) an element of $\mathbb{N} \times \mathbb{Z}/n\mathbb{Z}[X]^3 \times \mathbb{N}$ output by Gen and want to output a pair (x, y) of $((\mathbb{Z}/n\mathbb{Z})^\times)^2$, such that $P_1^{-1}(x) = P_2^{-1}(y)$. One way to do this is to pick some $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ and let $x = P_1(a)$ and $y = P_2(a)$. Intuitively, the *knowledge of preimage assumption* (KPA) can be viewed as saying that this is the "only" way to produce such a pair.

There are many ways in which the formulation can be varied to capture the KPA. We will say that for any PPTM outputting a pair $(P_1(a), P_2(a))$, there is an "extractor" than can return the preimage a. For our purposes, it is necessary

to allow the adversary to be randomized as in [1] (in that case, it is important that the extractor gets the coins of the adversary as an additional input, since otherwise the assumption is clearly false).

Definition 5. *Let* Gen *be a PDH generator and let* $\mathbf{A}$ *and* $\overline{\mathbf{A}}$ *be two PPTM's. We consider the following random experiments, where* $k \in \mathbb{N}$ *is a security parameter:*

Experiment $\mathbf{Exp}^{\mathsf{kpa}}_{\mathsf{Gen},\mathbf{A},\overline{\mathbf{A}}}(k)$

$(n, p, q, P_1, P_2, R) \xleftarrow{R} \mathsf{Gen}(k)$

$(x, y) \xleftarrow{r} \mathbf{A}(n, P_1, P_2)$

$\alpha \xleftarrow{r} \overline{\mathbf{A}}(n, P_1, P_2)$

Return 1 *if* $(x, y) \in ((\mathbb{Z}/n\mathbb{Z})^\times)^2$, $\exists a \in (\mathbb{Z}/n\mathbb{Z})^\times$ *s.t.*

$(x, y) = (P_1(a), P_2(a))$ *and* $a \neq \alpha$,

Return 0 *otherwise*

We define the advantage *of* $\mathbf{A}$ *relative to* $\overline{\mathbf{A}}$ *in these experiments* via

$$\mathbf{Adv}^{\mathsf{kpa}}_{\mathsf{Gen},\mathbf{A},\overline{\mathbf{A}}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{kpa}}_{\mathsf{Gen},\mathbf{A},\overline{\mathbf{A}}}(k) = 1\right].$$

Let $\varepsilon \in [0,1]^{\mathbb{N}}$,

1. $\overline{\mathbf{A}}$ *is a* ε-kpa-extractor *for* $\mathbf{A}$ *if for all positive integers* k,

$$\mathbf{Adv}^{\mathsf{kpa}}_{\mathsf{Gen},\mathbf{A},\overline{\mathbf{A}}}(k) \leq \varepsilon(k).$$

2. *We say that the knowledge-of-preimage assumption holds for* Gen *if for every PPTM* $\mathbf{A}$*, there exists a PPTM* $\overline{\mathbf{A}}$ *and a negligible function* ε *such that* $\overline{\mathbf{A}}$ *is a* ε-KPA-extractor *for* $\mathbf{A}$.
3. *We say that the strong knowledge-of-preimage assumption (SKPA) holds for* Gen *if there exists a PPTM* $\mathcal{E}$ *such that for every PPTM* $\mathbf{A}$*, there exists a negligible function* ε *such that* $\mathcal{E}$ *is a* ε-KPA-extractor *for* $\mathbf{A}$.

6.3 New Construction

Following Damgård's technique, we define a new encryption scheme where the public key is (n, P_1, P_2, Q) where P_1, P_2 and Q are One-Way permutations of $(\mathbb{Z}/n\mathbb{Z})^\times$ and the corresponding secret key is P^{-1}. To encrypt a message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$, a user picks at random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$ and uses the following encryption function:

Function 1: $(m, r) \mapsto \big(P_1(r), P_2(r), m \cdot Q(r)\big)$

When he receives a ciphertext (x, y, C), a user uses his knowledge of P^{-1} to check whether the equality $P_2(P_1^{-1}(x)) = y$ holds: if it is the case, he retrieves the message m, as $m = C/Q(P_1^{-1}(x))$, otherwise he rejects the ciphertext.

Theorem 10. *The previous schemes is One-Way and semantically secure under non-adaptive Chosen Ciphertext Attack relative to the following problems:*

Encryption function	*One-Wayness*	*Semantic security*
Function *1*	C-DPOL(n, P_1, Q)	D-DPOL(n, P_1, Q) *under SKPA*

Proof (Sketch). Theorem 8 insures that this scheme is one-way assuming the intractability of the C-DPOL(n, P_1, Q) problem and semantically secure under chosen-plaintext attacks if the D-DPOL(n, P_1, Q) problem is intractable. Following [4], it is straightforward to see that the scheme is plaintext-aware (PA1) assuming the strong knowledge of preimage assumption and *Theorem 1* from this paper implies that our new scheme is IND-CCA1-secure if the D-DPOL(n, P_1, Q) problem is intractable and the strong knowledge of preimage assumption holds for the underlying PDH generator. □

If one sets $P_1(X) = X^e$, $P_2(X) = (X+1)^e$ and $Q(X) = (X+2)^e$ with sufficiently large e in order to make the D-DPOL problem infeasible in reasonable time (see the analysis of subsection 3.2 and section 4) one obtains an IND-CCA1-secure system faster than the one of Damgård.

7 Conclusion

We have defined new algorithmic problems, derived from the RSA assumption, and discuss their computational difficulty. We have applied them to design public key encryption protocols with IND-CPA-security and IND-CCA2-security in the random oracle model under the assumption of the intractability of their decisional variants.

The ideas developed in this extended abstract can be used to design encryption schemes with higher security. For instance, by using the approach proposed by Cramer and Shoup in [9], we have been able to design a concrete encryption scheme that is proven IND-CCA2-secure in the standard model based on the difficulty of the new algorithmic problems. Details will appear elsewhere.

References

1. Barak, B., Lindell, Y., Vadhan, S.: Lower Bounds for Non-Black-Box Zero Knowledge.. In: Sudan, M. (ed.) Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003), pp. 384–393. IEEE Computer Society, Los Alamitos (2003)
2. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) [18], pp. 26–45
3. 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)

4. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
5. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Denning, D., Pyle, R., Ganesan, R., Sandhu, R., Ashby, V. (eds.) Proceedings of the First ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
6. Castagnos, G.: An efficient probabilistic public-key cryptosystem over quadratic fields quotients. Finite Fields Appl. 13(3), 563–576 (2007)
7. Catalano, D., Gennaro, R., Howgrave-Graham, N., Nguyen, P.Q.: Paillier's cryptosystem revisited. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 206–214 (2001)
8. Coppersmith, D., Franklin, M., Patarin, J., Reiter, M.: Low-Exponent RSA with Related Messages. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 1–9. Springer, Heidelberg (1996)
9. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack. SIAM J. Comput. 33(1), 167–226 (2003)
10. Damgård, I.B.: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In: Brickell, E.F. (ed.) CRYPTO 1991. LNCS, vol. 740, pp. 445–456. Springer, Heidelberg (1993)
11. De Marchi, S.: Polynomials arising in factoring generalized Vandermonde determinants: an algorithm for computing their coefficients. Math. and Comput. Modelling 34(3–4), 271–281 (2001)
12. Demytko, N.: A Elliptic Curve Based Analogue of RSA. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 40–49. Springer, Heidelberg (1994)
13. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Trans. Inf. Theory 22, 644–654 (1976)
14. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, Cambridge (1999)
15. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. 28, 270–299 (1984)
16. Hada, S., Tanaka, T.: On the Existence of 3-Round Zero-Knowledge Protocols. In: Krawczyk, H. (ed.), [18], pp. 408–423
17. Joye, M., Quisquater, J.: Efficient computation of full Lucas sequences. Electronics Letters 32(6), 537–538 (1996)
18. Krawczyk, H. (ed.): CRYPTO 1998. LNCS, vol. 1462. Springer, Heidelberg (1998)
19. Lidl, R., Mullen, G.L., Turnwald, G.: Dickson Polynomials., Pitman Monographs and Surveys in Pure and Applied Mathematics, vol. 65. Longman Scientific & Technical, New York (1993)
20. Müller, W.B., Nöbauer, R.: Some remarks on public-key cryptosystems. Sci. Math. Hungar 16, 71–76 (1981)
21. Müller, W.B., Nöbauer, R.: Cryptanalysis of the Dickson-scheme. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 50–61. Springer, Heidelberg (1985)
22. Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
23. Pointcheval, D.: New Public Key Cryptosystems Based on the Dependent-RSA Problems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 239–254. Springer, Heidelberg (1999)

24. Rivest, R.L., Shamir, A., Adleman, L.M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Comm. ACM 21, 120–126 (1978)
25. Schwenk, J., Huber, K.: Public key encryption and digital signatures based on permutation polynomials. Electronics Letters 34(8), 759–760 (1998)
26. Smith, P., Lennon, M.J.J: LUC: A new public key system. In: Proc. of the Ninth IFIP Int. Symp. on Computer Security, pp. 103–117 (1993)

A Generalization and a Variant of Two Threshold Cryptosystems Based on Factoring

Yvo Desmedt[1] and Kaoru Kurosawa[2]

[1] Department of Computer Science, University College London, UK
y.desmedt@cs.ucl.ac.uk
[2] Department of Computer and Information Sciences, Ibaraki University, Japan
kurosawa@mx.ibaraki.ac.jp

Abstract. At Asiacrypt 2002, Katz and Yung presented two threshold cryptosystems based on factoring, a threshold version of Goldwasser-Micali's probabilistic encryption assuming that $p = q = 3 \bmod 4$, and a threshold Rabin signature scheme assuming that $p = 3 \bmod 8$ and $q = 7 \bmod 8$. In this paper, we show a generalized condition on p and q to obtain a threshold version of Goldwasser-Micali, and a threshold Rabin-type signature scheme due to Kurosawa and Ogata [7] for $p = q = 3 \bmod 4$ and

$$\frac{p+1}{4} = \frac{q+1}{4} \bmod \gcd(p-1, q-1).$$

Note that our set of (p, q) is disjoint from that of Katz-Yung threshold Rabin signature scheme.

Keywords: Threshold signatures, threshold decryption, Goldwasser-Micali, Rabin, cryptography.

1 Introduction

Katz and Yung [8] showed threshold versions of two cryptosystems based on factoring $N = pq$, where p and q are large primes.

1. First they considered Goldwasser-Micali's probabilistic encryption scheme [5] and gave a threshold decryption scheme assuming that that $p = q = 3 \bmod 4$.
2. Second, they presented a threshold Rabin's signature scheme assuming that $p = 3 \bmod 8$ and $q = 7 \bmod 8$.

In this paper, we show

1. a generalized condition on p and q to obtain a threshold version of Goldwasser-Micali and
2. a threshold Rabin-type signature scheme due to Kurosawa and Ogata [7] for $p = q = 3 \bmod 4$ and

$$\frac{p+1}{4} = \frac{q+1}{4} \bmod \gcd(p-1, q-1).$$

Note that our set of (p, q) is disjoint from that of Katz-Yung scheme.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 351–361, 2007.

We now motivate the importance of this research. Several problems are known to be self reducible. This property has been used to study average complexity in contrast to worst case studies (as NP-completeness). However, factoring is not known to be self reducible. So, it is possible that for certain families of N, the product of two primes, factoring is easier than for others. So far, no such evidence has been found. However, before Lenstra's elliptic curve factoring algorithm [9] it was unknown that factoring integers with relatively small primes was easier than factoring other integers. Evidently it was known that the complete factorization of smooth numbers (numbers with only very small primes) was trivial.

For the aforementioned reason it is important that when using factoring *based* primitives, such as quadratic residuosity, one has schemes available that work for an as large as possible families of N. We therefore revisit Katz-Yung with the goal to extend their work for families of N for which their schemes failed.

For the reader not familiar with threshold cryptography, we recommend the reading of [8], which has an extensive bibliography.

Our paper is organized as following. We first present some preliminaries. We then discuss the new threshold signature scheme for a variant of Rabin's signature scheme. Finally we explain how to use classical results on threshold cryptography to achieve threshold decryption of Goldwasser-Micali.

2 Preliminaries

2.1 Notation and Definitions

Let

$$QR_N = \{x \mid x = a^2 \bmod N \text{ for some } a \in Z_N^*\}.$$

If $N = pq$ and p, q are two primes, then $\phi(N) = (p-1)(q-1)$ and $\lambda(N) = \text{lcm}(p-1, q-1)$ as usual.

For a prime p, $\left(\frac{x}{p}\right)$ denotes Legendre symbol. For a composite N, $\left(\frac{x}{N}\right)$ denotes Jacobi symbol. If $p = 3 \bmod 4$, then $\left(\frac{-1}{p}\right) = -1$.

2.2 Threshold Signature Scheme

We consider a (k, n)-threshold signature scheme. (For more background information on threshold cryptography, see e.g. [8].) At the beginning of the game, the adversary selects a subset of $k-1$ players to corrupt. Hence we consider static adversaries (i.e., during the protocol the set of adversaries cannot be modified).

In the dealing phase, the dealer generates a public-key pk along with secret-key shares $sk_1, \cdots, sk_n$. The adversary obtains the secret-key shares of the corrupted players along with the public-key.

After the dealing phase, the adversary submits signing requests to the uncorrupted players for messages of his choice. Upon such a request, each player outputs a signature share for the given message.

The share combining algorithm takes as input a message and k valid signature shares on the message along with the public-key, and outputs a valid signature on the message.

We say that the adversary A forges a signature if at the end of the game, he outputs a valid signature on a message that was not submitted as a signing request to the uncorrupted players. A succeeds if A forges a signature.

We say that the threshold signature scheme is secure against chosen message attack if there exists no probabilistic polynomial time adversary A who forges a signature with nonnegligible probability.

2.3 Threshold Decryption Scheme

We consider a (k, n)-threshold decryption scheme. At the beginning of the game, the adversary selects a subset of $k-1$ players to corrupt.

The dealing phase is similar to the one for threshold signing (see Section 2.2).

Proving the security of threshold decryption schemes against chosen ciphertexts is quite complex. So, we only focus on semantic security. After the dealing phase, the adversary observes the threshold decryption of ciphertexts. Upon such a request, each player outputs a decryption share for the given ciphertext.

The combining algorithm takes as input the ciphertext and k valid decryption shares on the ciphertext along with the public-key, and outputs a valid plaintext message.

We say that the adversary A succeeds if at the end of the game, he outputs the plaintext of a ciphertext that was chosen before the start of the game and not submitted as a decryption request to the uncorrupted players.

We say that the threshold decryption scheme is semantically secure if there exists no probabilistic polynomial time adversary A who succeeds with nonnegligible probability (for a more formal definition see the literature on threshold decryption).

2.4 KO Variant of Rabin's Signature Scheme

In the original Rabin's signature scheme, a signature on a message m is given by (σ, R) such that

$$H(m, R) = \sigma^2 \bmod N,$$

where H is a hash function and R is a random string such that $H(m, R) \in QR_N$. In this scheme, however, a random string R is included in the signature. It is known that we can avoid this problem if $p = 3 \bmod 8$ and $q = 7 \bmod 8$.

Kurosawa and Ogata showed a variant of Rabin's signature scheme such that p and q are arbitrary which can still eliminate R [7]. Their scheme is secure against chosen message attack in the random oracle model if factoring $N = pq$ is hard [7].

Definition 1. *Let $N = pq$, where p and q are primes. For $x \in Z_N^*$, let*

$$u = \left(\frac{x}{p}\right), \ v = \left(\frac{x}{q}\right).$$

Define

$$type(x) \stackrel{\triangle}{=} \begin{cases} 0 & if \ \ u = v = 1 \\ 1 & if \ \ u = 1, v = -1 \\ 2 & if \ \ u = -1, v = 1 \\ 3 & if \ \ u = v = -1 \end{cases} \tag{1}$$

Proposition 1. *$xy \in QR_N$ if and only if $type(x) = type(y)$.*

Key Generation. The signer chooses two ℓ-bit primes p and q, and sets $N = pq$. Next, the signer chooses α_i such that $type(\alpha_i) = i$ for $i = 0, \cdots, 3$ randomly.

If $p = q = 3 \bmod 4$, this can be done as follows. The signer first chooses α_1 such that $type(\alpha_1) = 1$ randomly. Next let $\alpha_0 = 1, \alpha_2 = -\alpha_1, \alpha_3 = -1 \bmod N$.

Finally, let $H : \{0,1\}^* \rightarrow Z_N^*$ be a hash function, and set the verification key as $(N, H, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$ and the signing key as (p, q).

Signing. For a message m , the signer executes the following steps.

Step 1: Compute i such that $type(H(m)) = i$.

Step 2: For this i, compute $\sigma \in QR_N$ such that

$$\alpha_i H(m) = \sigma^2 \bmod N \tag{2}$$

The signature is (i, σ).

Note that $\alpha_i H(m) \in QR_N$ from Proposition 1.

Verification. On input a message m and a signature (i, σ), the verifier checks if Eq.(2) is satisfied.

3 Proposed Rabin-Type Threshold Signature Scheme

3.1 Idea

Suppose that two primes p and q satisfy $p = q = 3 \bmod 4$ and

$$\frac{p+1}{4} = \frac{q+1}{4} \bmod \gcd(p-1, q-1) \tag{3}$$

Let $N = pq$ for such p, q.

Lemma 1. *Let $d \in Z_{\lambda(N)}$ be such that*

$$d = (p+1)/4 \bmod (p-1) \tag{4}$$

$$d = (q+1)/4 \bmod (q-1) \tag{5}$$

Then we can compute a square root of $a \in QR_N$ as

$$x = a^d \bmod N.$$

(Proof) First Eq.(4) and Eq.(5) have a solution because Eq.(3) is satisfied.

Next if $p = 3 \bmod 4$, then it is known that $x = a^{(p+1)/4} \bmod p$ is a square root of $a \in QR_p$. Similarly, $x = a^{(q+1)/4} \bmod q$ is a square root of $a \in QR_q$. Hence $x = a^d \bmod N$ is a square root of $a \in QR_N$. Q.E.D.

Lemma 2. *The above x satisfies $x \in QR_N$.*

(Proof) This is because

$$\left(\frac{x}{p}\right) = \left(\frac{a^d}{p}\right) = \left(\frac{a}{p}\right)^d = 1^d = 1$$
$$\left(\frac{x}{q}\right) = \left(\frac{a^d}{q}\right) = \left(\frac{a}{q}\right)^d = 1^d = 1 \qquad Q.E.D.$$

In our threshold scheme, the dealer distributes the above d to the players. If $type(H(m)) = 0$, then the players jointly compute $\sigma = H(m)^d \bmod N$. However, the players cannot compute $type(H(m))$ because they do not know p, q. We solve this problem by using the following lemma.

Lemma 3. *Suppose that $\left(\frac{a}{N}\right) = 1$. Let $x = a^d \bmod N$. Then*

$$x^2 = a \bmod N \text{ or } x^2 = -a \bmod N.$$

(Proof) Since $\left(\frac{a}{N}\right) = 1$, $a \in QR_N$ or $-a \in QR_N$. First suppose that $a \in QR_N$. Then $x^2 = a \bmod N$ from Lemma 1.

Next suppose that $-a \in QR_N$. Then since $-a \in QR_p$, we have

$$\left(\frac{a}{p}\right) = -1 \text{ because } 1 = \left(\frac{-a}{p}\right) = -\left(\frac{a}{p}\right).$$

Now it holds that

$$x^2 = a^{2d} = a^{(p+1)/2} = a^{\frac{p-1}{2}+1} = a^{(p-1)/2}a = -a \bmod p.$$

Similarly, we have $x^2 = -a \bmod q$. This means that $x^2 = -a \bmod N$. Q.E.D.

3.2 A *k*-Out-of-*k* Protocol

We present the k-out-of-k protocol in this subsection.

Key Generation

1. The dealer generates two primes p and q which satisfy $p = q = 3 \bmod 4$ and Eq.(3). He sets $N = pq$. He chooses $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ as shown in Sec.2.4. Let $H : \{0,1\}^* \to Z_N^*$ be a hash function. The public-key of the protocol is $(N, H, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$.

2. The dealer computes $d \in Z_{\lambda(N)}$ which satisfies Eq.(4) and Eq.(5). He chooses $d_1, \cdots, d_k \in Z_{\phi(N)}$ randomly such that

$$d_1 + \cdots + d_k = d \bmod \phi(N).$$

3. Finally, the dealer sends d_u to player u as a secret-key share for $u = 1, \cdots, k$.

Signing. For a message m, the players first publicly compute $\left(\frac{H(m)}{N}\right)$.

– If $\left(\frac{H(m)}{N}\right) = 1$, then each player u outputs a signature share

$$y_u = (H(m))^{d_u} \bmod N.$$

The combining algorithm computes σ as follows.

$$\sigma = y_1 \times \cdots \times y_k \bmod N.$$

It then checks if $\sigma^2 = H(m)$ or $-H(m) \bmod N$. If $\sigma^2 = H(m)$, then it outputs a signature $(0, \sigma)$. Otherwise, it outputs a signature $(3, \sigma)$.

– If $\left(\frac{H(m)}{N}\right) = -1$, then each player u outputs a signature share

$$y_u = (\alpha_1 H(m))^{d_u} \bmod N.$$

The combining algorithm computes σ as follows.

$$\sigma = y_1 \times \cdots \times y_k \bmod N.$$

If $\sigma^2 = \alpha_1 H(m)$, then it outputs a signature $(1, \sigma)$. Otherwise, it outputs a signature $(2, \sigma)$.

Theorem 1. *The above protocol is secure against chosen message attack for any passive adversary, assuming the hardness of factoring (in the random oracle model).*

(Proof) Suppose that there exists a probabilistic polynomial time adversary A who forges a signature with nonnegligible probability. We show a simulator S who mounts a chosen message attack on the KO signature scheme. Wlog, assume that an adversary A corrupts players $1, \cdots, k-1$.

Dealing Phase
Input to S: $N(= pq)$ and a hash function $H : \{0,1\}^* \to Z_N^*$, where two primes p and q satisfy $p = q = 3 \bmod 4$ and Eq.(3).

1. S chooses α_1 randomly such that $\left(\frac{\alpha_1}{N}\right) = -1$. S sets $\alpha_0 = 1, \alpha_2 = -\alpha_1$ and $\alpha_3 = -1$. S gives $(N, H, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$ to A as a public-key.
2. S chooses $d_1, \cdots, d_{k-1}$ such that $d_i \in \{0, 1, \cdots, N - 2\sqrt{N}\}$ randomly, and gives them to A as secret-key shares.

Signature generation phase
Suppose that A submits a signing request for a message m to the uncorrupted player k. S queries m to her sign oracle. Suppose that the oracle returns a signature (Index, σ).

– If $\left(\frac{H(m)}{N}\right) = 1$, then S computes y_k such that

$$\sigma = (H(m))^{d_1} \times \cdots \times (H(m))^{d_{k-1}} \times y_k \bmod N.$$

– If $\left(\frac{H(m)}{N}\right) = -1$, then S computes y_k such that

$$\sigma = (\alpha_1 H(m))^{d_1} \times \cdots \times (\alpha_1 H(m))^{d_{k-1}} \times y_k \bmod N.$$

S gives d_k to A as the signature share of player k. Finally, A outputs a forgery $(\tilde{m}, (\tilde{i}, \tilde{\sigma}))$. S then A outputs $(\tilde{m}, (\tilde{i}, \tilde{\sigma}))$ as her forgery.

Now the distribution on $(d_1, \cdots, d_{k-1})$ is statistically indistinguishable from the real distribution, and $type(\alpha_1) = 1$ with probability $1/2$. S simulates the other part of the environment of A perfectly. Hence A outputs a forgery with nonnegligible probability. Therefore, S also outputs a forgery with nonnegligible probability. However, it is known that KO scheme is secure against chosen message attack by assuming the hardness of factoring [7]. This is a contradiction.

Hence the above protocol is secure against chosen message attack for passive adversaries. Q.E.D.

3.3 Extensions

We can construct a (k, n)-threshold scheme and achieve robustness by using the same technique as shown in [8, Sec.3.2].

For example, by using the idea of [11], our (n, n)-threshold signature scheme can be converted to a (k, n)-threshold one as follows.

Key Generation

1. The dealer generates $(P, N, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$ and $d_1, \cdots, d_n$ as shown in Sec.3.2.
2. The dealer chooses a prime $P > N$ as well. The public-key is $(P, N, \alpha_0, \alpha_1, \alpha_2, \alpha_3)$.
3. For each d_i, the dealer chooses a random $(k-1)$-degree polynomial f_i over Z_P such that $f_i(0) = d_i$.
4. To player j, the dealer sends d_j and $f_i(j)$ for $1 \leq i \leq n$.

Signing

1. Each player u broadcasts y_u as shown in Sec.3.2.
2. If player i cannot participate, the remaining players reconstruct d_i using the shares $f_i(j)$s. By using d_i, the y_i is computed.
3. Finally σ is computed as shown in Sec.3.2.

Robustness is obtained by applying the technique given in [8, Sec.3.2].

4 Threshold Decryption of Goldwasser-Micali

4.1 Idea

The main idea behind Katz-Yung threshold decryption scheme of Goldwasser-Micali, is the following observation. Suppose that $N = pq$, where $p = q = 3 \bmod 4$ are primes. Then the condition to check whether C, with Jacobi symbol 1, is a quadratic residue or not, reduces to checking whether $C^{(N-p-q+1)/4} = 1 \bmod N$ [8]. This allowed Katz-Yung to use exponentiation in their threshold decryption.

When $p = 1 \bmod 4$ or $q = 1 \bmod 4$, this test to check whether C is quadratic residue modulo N, no longer works. Elementary number theory however allows us to find another test, for several cases. We now elaborate.

4.2 Using Number Theory

Euler's criterion tells us that when p is an odd prime

$$\left(\frac{C}{p}\right) = C^{\frac{(p-1)}{2}} \bmod p.$$

So, to allow us to continue to use the power of exponentiation to achieve threshold decryption, we wonder whether there exists an exponent a such that $C^a \bmod N$ returns 1 if $C \in QR_N$ and otherwise -1 when C has Jacobi symbol 1. Evidently, such an a exists if

$$\begin{aligned} a &= (p-1)/2 \bmod p-1 \\ a &= (q-1)/2 \bmod q-1 \end{aligned} \tag{6}$$

The generalized Chinese Remainder Theorem tells us, this a exists modulo $\lambda(N)$ if and only if $\gcd(p-1, q-1) \mid (p-1)/2 - (q-1)/2$, or

$$\gcd(p-1, q-1) \mid (p-q)/2. \tag{7}$$

We now distinguish between the following cases:

- $p = 3 \bmod 4$ and $q = 1 \bmod 4$. In this case $p = 4p' + 3$ and $q = 4q' + 1$, where p' and q' are not necessarily primes. $(p-q)/2 = 2(p'-q') + 1$. However, $\gcd(p-1, q-1)$ is even. So, the condition will always be violated.
- $p = 1 \bmod 4$ and $q = 3 \bmod 4$. This case is similar as above.
- $p = 3 \bmod 4$ and $q = 3 \bmod 4$. Let $p = 4p' + 3$ and $q = 4q' + 3$, where p' and q' are not necessarily primes. $(p-q)/2 = 2(p'-q')$. It is now easy to see the condition in Eq. 7 is always satisfied. [1]
- $p = 1 \bmod 4$ and $q = 1 \bmod 4$. Let $p = 4p' + 1$ and $q = 4q' + 1$, where p' and q' are not necessarily primes. It is not too difficult to see that there are cases for p' and q' for which Eq. 7 will be satisfied and others for which it will not.

[1] Note that $\gcd(p-1, q-1)$ is not divisible by 4.

We now analyze the case $p = 1 \bmod 4$ and $q = 1 \bmod 4$ where p' and q' are as above in more details. Note that $(p-q)/2 = 2(p'-q')$ and $\gcd(p-1, q-1)$ is this time divisible by 4. Hence

- If both p' and q' are odd, Eq. 7 will be satisfied.
- If p' is even and q' is odd (or vice versa), Eq. 7 will *not* be satisfied.
- Consider the case both p' and q' are even. Now write $p' = 2p''$ and $q' = 2q''$. It is now easy to see that the analysis is similar, but replacing p' by p'' and q' by q'' in the argument.

This brings us easily to the condition that if

$$p = q = 2^k + 1 \bmod 2^{k+1} \qquad \text{where } k \geq 2$$

Eq. 7 is satisfied.

4.3 A Generic Scheme

When Eq. 7 is satisfied, the Chinese Remainder Theorem allows us to compute an $a \bmod \lambda(N)$, which allows us to proceed with a threshold decryption of Goldwasser-Micali similarly as the threshold decryption in RSA. We now explain the details.

Decryption in the original Goldwasser-Micali scheme corresponds to decide whether the ciphertext C (with Jacobi symbol 1) is a quadratic residue or not. In Goldwasser-Micali's scheme this can easily be done by the receiver of the message, since this one knows the factorization of N. In a threshold decryption scheme, the multiple parties (at least k) that will help the receiver in decrypting the ciphertext should obviously not learn this factorization. Katz-Yung observed that when $N = pq$ and $p = q = 3 \bmod 4$ checking whether C is a quadratic residue or not corresponds to an exponentiation modulo N. The most famous scheme in cryptography that uses exponentiation modulo a composite is RSA.

Decryption in RSA is done by raising the ciphertext by a secret exponent d modulo N. For RSA, several threshold schemes have been developed (consult [4] for a survey until 1997, and e.g. [2]). In all these threshold RSA schemes, the secret d is shared in a preliminary phase, without revealing anything about the factorization of N. Different techniques are then used to compute in a shared way $C^d \bmod N$. Depending on the scheme used the security against insiders could be limited to being passive, or active. The last case is more difficult to deal with (see also [1]). To decrypt the ciphertext, the parties perform a shared exponentiation. Typically, the results of these are sent to the receiver, who acts as a combiner by typically performing some multiplications modulo N. The receiver does not have any secrets.

We now explain threshold Goldwasser-Micali decryption. The method is generic in the sense that some schemes for RSA threshold decryption can be used for threshold Goldwasser-Micali decryption as shown in [8]. We now describe the details. When we refer to a share distribution scheme, a shared exponentiation scheme, and a combiner scheme we refer to these coming from RSA threshold decryption scheme.

If $N = pq$, where $p = q = 2^k + 1 \bmod 2^{k+1}$ and $k \geq 2$, checking whether C is a quadratic residue or not, can be done by raising C to a, where a is as in Eq. 6. To achieve threshold decryption, in a preliminary phase, the parties receive shares of a based on the share distribution scheme. To decrypt a ciphertext C they proceed using the threshold exponentiation scheme. Each party sends its result to the receiver. The receiver performs the combiner scheme and obtains $C^a \bmod N$, which is 1 or -1. This immediately implies the plaintext.

4.4 The Security

Since Goldwasser-Micali's scheme provides only semantic security, the threshold variant also does not deal with active adversaries. The security proofs follow immediately from the one of Goldwasser-Micali and the security proof of the underlying threshold RSA (threshold exponentiation modulo a composite) scheme.

Acknowledgment

The first author is BT Professor of Information Security at University College London, Adastral Park and courtesy professor at Florida State University, USA. Partial funding provided for the first author by EPSRC EP/C538285/1.

The first author e-mailed Yair Frankel with suggestions for threshold decryption of Goldwasser-Micali on February 22, 1995.

The authors thank the referees for their comments.

References

1. Canetti, R., Goldwasser, S.: An Efficient Threshold Public Key Cryptosystem. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90–106. Springer, Heidelberg (1999)
2. Cramer, R., Fehr, S.: Optimal black-box secret sharing over arbitrary abelian groups. Crypto 2002, 272–287 (2002)
3. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, Springer, Heidelberg (1998)
4. Desmedt, Y.: Some recent research aspects of threshold cryptography. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 158–173. Springer, Heidelberg (1998)
5. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
6. Jarecki, S., Lysyanskaya, A.: Adaptively Secure Threshold Cryptography. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 221–242. Springer, Heidelberg (2000)
7. Kurosawa, K., Ogata, W.: Efficient Rabin-type digital signature scheme. Design, Codes and Cryptography 16(1), 53–64 (1999)
8. Katz, J., Yung, M.: Threshold Cryptosystems Based on Factoring. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 192–205. Springer, Heidelberg (2002)

9. Lenstra Jr., H.W.: Factoring integers with elliptic curves. Annals of Mathematics 126, 649–673 (1987)
10. Rabin, M.: Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology Technical Report MIT/LCS/TR–212, Cambridge, Massachusetts (January 1977)
11. Rabin, T.: A Simplified Approach to Threshold and Proactive RSA. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 89–104. Springer, Heidelberg (1998)

Towards a DL-Based Additively Homomorphic Encryption Scheme

Guilhem Castagnos[1] and Benoît Chevallier-Mames[2]

[1] DMI-XLIM,
Université de Limoges,
123, Avenue Albert-Thomas,
87060 Limoges Cedex, France
guilhem.castagnos@unilim.fr
[2] Gemalto, Security Labs,
La Vigie, Avenue du Jujubier, ZI Athélia IV,
F-13705 La Ciotat Cedex, France
benoit.chevallier-mames@gemplus.com

Abstract. ElGamal scheme has been the first encryption scheme based on discrete logarithm. One of its main advantage is that it is simple, natural and efficient, but also that its security is clearly understood. However, one of its — often forgotten — disadvantages is that this scheme requires the encoding of messages into group elements, in order to be semantically secure. Unfortunately, this need prevents the scheme to be fully practical.

In this paper, we propose a new way to deal with the problem of message encoding, which offers several advantages though some disadvantages. Our scheme is based on a quite simple combination of the standard ElGamal scheme with a message encoding inspired by the Naccache-Stern cryptosystem. We consider our solution as a new step towards the open problem of designing a discrete-logarithm based encryption scheme with the property of being additively homomorphic. Unfortunately, our construction is still not a complete solution. We hope however that it might give clues for a possible full solution.

Keywords: ElGamal encryption scheme, Naccache-Stern cryptosystem, DL-based homomorphic scheme, standard model, public-key cryptography.

1 Introduction

Since the discovery of public-key cryptography by Diffie and Hellman [DH76], several encryption schemes have been proposed, but very few of them had real impact on the academic community or industry. Clearly, it is commonly agreed that RSA [RSA78] and ElGamal [ElG85] are of this kind.

More precisely, ElGamal scheme has been the first encryption scheme based on discrete logarithm. One of its main advantage is that it is simple, natural and

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 362–375, 2007.

efficient, but also that its chosen-plaintext security is clearly understood: under the so-called CDH assumption, the one-wayness is ensured; under the so-called DDH assumption, the scheme is semantically secure. However, one of its often forgotten disadvantages is that this scheme requires the encoding of messages into group elements, to ensure indistinguishability. Unfortunately, this need prevents the ElGamal scheme to be fully practical, and its homomorphic properties to be really used.

To get rid off this problem, either the so-called hashed-ElGamal is preferred (in which case, the security is only ensured in the random oracle model), or the construction is totally modified. Note that the Cramer-Shoup encryption scheme (cf. [CS98]), whose IND-CCA proof is valid in the standard model, also requires this encoding.

On the contrary to the problem of designing additive homomorphic encryption schemes based on factorization, which has already been efficiently solved by Paillier [Pai99], after other less-efficient constructions such as Goldwasser-Micali [GM84] or Okamoto-Uchiyama [OU98], no practical homomorphic DL-based primitive is currently known. One would note that the DL-type encoding-free scheme proposed by Chevallier-Mames, Paillier and Pointcheval [CPP06] offers a very-weak kind of malleability, in the sense that one can add a plaintext to a ciphertext without decrypting.

These malleability properties (which also include self-blinding, *i. e.*, the ability of "re-randomizing" a ciphertext) are of great use in certain applications such as e-votes or banking. For example the system of Paillier, and its generalization proposed by Damgård and Jurik [DJ01], have been used to design electronic vote systems [BFP+01, Jur03], for Private Information Retrieval [Lip05], or for building Mix-nets [NSNK06, Jur03].

Our Contribution. In this paper, we propose a new solution to the encoding problem, with a security (for chosen-plaintext attacks) in the standard model, under classical assumptions (namely, the DDH assumption). As we explain later, our scheme offers several advantages, though some disadvantages.

Roughly, our scheme is a simple and natural combination of ElGamal with a message encoding inspired by the Naccache-Stern [NS97] cryptosystem. In this regard, and even if our solution has been designed and studied mainly because of the encoding difficulties, our scheme can also be seen as a modification of the original Naccache-Stern construction in order to achieve a certain proof of security: indeed, nothing is really known about the plain Naccache-Stern scheme.

Last but not least, our solution might be seen as a new step towards the design of an additive homomorphic encryption scheme based on the discrete logarithm problem: under conditions that will be detailed, addition of ciphertexts is possible. Moreover, as we do not change the construction of ElGamal, our solution still offers full self-blinding. We hope that our construction might give clues for a possible future full solution.

Outlines. Our paper is organized as follows. In the second part, we remind the background needed for this work, notably definitions of encryption schemes and of their security notions. Then, we describe the ElGamal encryption schemes, and why encoding is needed. In the same section, we also expose solutions that were already proposed to deal with this inherent problem. Fourth part is the main part of our paper: it describes our new encoding for the ElGamal scheme and details its features, efficiency and malleability properties. Finally, we conclude our paper, by opening new problems that are consequences of our work.

2 Preliminaries

In this section, we briefly remind the background regarding public key encryption.

2.1 Public-Key Encryption

We describe a public-key encryption scheme $\mathcal{E}$ as four probabilistic algorithms, $\mathcal{E} = (\text{Set}_{\mathcal{E}}, \text{Gen}_{\mathcal{E}}, \text{Encrypt}, \text{Decrypt})$:

Setup. Given a security parameter k, $\text{Set}_{\mathcal{E}}(1^k)$ produces some common parameters params, used by the three others algorithms.

Key Generation. Given a security parameter k, $\text{Gen}_{\mathcal{E}}(1^k)$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of public and private keys.

Encryption. Given a message m and a public key pk, $\text{Encrypt}_{\mathsf{pk}}(m)$ produces a ciphertext c. As for security reasons, the procedure is typically probabilistic, we write $c = \text{Encrypt}_{\mathsf{pk}}(m, r)$ where r denotes the randomness used by Encrypt.

Decryption. Given a ciphertext c and a private key sk, $\text{Decrypt}_{\mathsf{sk}}(c)$ returns a plaintext m or a special symbol $\perp$ if the ciphertext is invalid.

We will say that a public-key encryption scheme $\mathcal{E}$ is *additively homomorphic* if, given two ciphertexts $c_1 = \text{Encrypt}_{\mathsf{pk}}(m_1, r_1)$ and $c_2 = \text{Encrypt}_{\mathsf{pk}}(m_2, r_2)$ of unknown plaintexts m_1 and m_2, one can publicly compute a valid ciphertext c_3 of message $m_1 + m_2$.

Moreover, we will say that $\mathcal{E}$ allows *self-blinding*, if given c, an encryption of some (unknown) message m, it is possible to generate efficiently another unlinkable encryption c' of m.

2.2 Security Notions for Encryption Schemes

One-Wayness. The most important security notion that one would expect from an encryption scheme to fulfil is the property of *one-wayness* (OW): an attacker should not be able to recover the plaintext matching a given ciphertext. We capture this notion more formally by saying that for any adversary $\mathcal{A}$, succeeding in inverting the effects of Encrypt on a ciphertext c should occur with negligible probability. $\mathcal{A}$ is said to (k, ε, τ)-break OW when

$$\mathsf{Succ}^{\mathsf{OW}}_{\mathcal{E}}(\mathcal{A}) = \Pr_{m,r}[(\mathsf{pk},\mathsf{sk}) \leftarrow \textsc{Gen}_{\mathcal{E}}(1^k) \ : \ \mathcal{A}(\mathsf{pk}, \textsc{Encrypt}_{\mathsf{pk}}(m,r)) = m] \geq \varepsilon,$$

where the probability is taken over the random coins of the experiment and the ones of the adversary, and $\mathcal{A}$ halts after τ elementary steps. An encryption scheme is said to be one-way if no probabilistic algorithm (k,ε,τ)-breaks OW for $\tau \leq \mathsf{poly}(k)$ and $\varepsilon \geq 1/\mathsf{poly}(k)$.

Semantic Security. The notion of *semantic security* (IND) [GM84], as known as *indistinguishability of encryptions* captures a stronger notion of privacy. Here, the attacker should not learn any information whatsoever about a plaintext given its encryption. The adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is said to (k,ε,τ)-break IND when

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{E}}(\mathcal{A}) = 2 \times \Pr_{b,r}\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \textsc{Gen}_{\mathcal{E}}(1^k), (m_0,m_1,s) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ c = \textsc{Encrypt}_{\mathsf{pk}}(m_b,r) \ : \ \mathcal{A}_2(m_0,m_1,s,c) = b\end{array}\right] - 1 \geq \varepsilon,$$

where again the probability is taken over the random coins of the experiment as well as the ones the adversary. $\mathcal{A}$ must run in at most τ steps and it is imposed that $|m_0| = |m_1|$. An encryption scheme is said to be semantically secure or indistinguishable if no probabilistic algorithm can (k,ε,τ)-break IND for $\tau \leq \mathsf{poly}(k)$ and $\varepsilon \geq 1/\mathsf{poly}(k)$.

2.3 Computational Assumptions

We now briefly recall the definition of the discrete-log and related problems needed for the sake of this work. In what follows, $\mathbb{G}$ denotes an abelian finite group (denoted multiplicatively), described by a generator g and its prime order q.

Definition 1 (Discrete Logarithm – DL). *Given $g^x \in \mathbb{G}$ where $x \leftarrow \mathbb{Z}_q$, compute x.*

Definition 2 (Computational Diffie-Hellman – CDH). *Given $g^x \in \mathbb{G}$ and $g^y \in \mathbb{G}$ for $x, y \leftarrow \mathbb{Z}_q$, compute $g^{xy} \in \mathbb{G}$.*

Definition 3 (Decision Diffie-Hellman – DDH). *Let us consider the two distributions $D = (g^x, g^y, g^{xy})$ and $R = (g^x, g^y, g^z)$ for randomly distributed $x, y, z \leftarrow \mathbb{Z}_q$. Distinguish D from R.*

It is easily seen that $\mathsf{DDH} \Leftarrow \mathsf{CDH} \Leftarrow \mathsf{DL}$ where $\Leftarrow$ denotes polynomial reductions. In most cryptographic applications, the structure of the group $\mathbb{G}$ is chosen in such a way that these three computational problems seem intractable. A typical example is to choose $\mathbb{G} \subseteq \mathbb{Z}_p^*$ where q divides $(p-1)$ where classically, p is a 1024-bit prime and q a 160-bit prime. Another widely used family of groups is elliptic curves over large prime fields [Mil85, Kob87].

3 The ElGamal Cryptosystem

ElGamal encryption was introduced around twenty years ago [ElG85]. It requires a cryptographic group $\mathbb{G} = \langle g \rangle$ of order q. In the ElGamal scheme, one generates a public-private key pair by randomly selecting $x \leftarrow \mathbb{Z}_q$ and computing $y = g^x$. The public key is then y while the private key is x. In order to encrypt a message m, one randomly selects $r \leftarrow \mathbb{Z}_q$ and computes $u = g^r$ and $v = y^r \cdot m$. The ciphertext is $c = (u, v)$. Using the private key x, the ciphertext $c = (u, v)$ can be decrypted as $m = v \cdot u^{-x}$.

It is well-known that, for security reasons, one has first to use groups $\mathbb{G}$ of prime order, and second to define $\mathcal{M}$ as included in the group $\mathbb{G}$. Under these assumptions, it has been shown that ElGamal encryption is IND-CPA under the DDH assumption.

Unfortunately, the two above constraints make the ElGamal encryption scheme less practical than it appears at first sight. Indeed, before encryption takes place, the message must be *encoded* into a group element, and this group encoding must be efficiently invertible in order to allow the original message to be recovered during the decryption process. Such an encoding may be time consuming, and may also partially or totally destroy the inherent homomorphic property of the system. Also, using a group encoding remains incompatible with the optimization which consists in working in a small subgroup, $\mathbb{G}$, of $\mathbb{Z}_p^\star$ of prime order q where q is a 160-bit prime, a setting in which group exponentiations are much faster. Indeed, the only known way to preserve the homomorphic property is to use a morphism from $\mathbb{Z}_q$ to $\mathbb{G}$, such as $m \mapsto g^m$, as an encoding function. Unfortunately, this leads to an inefficient decryption process, as one has to compute a discrete logarithm to reverse the encoding (see the computation of the tally in [CGS97] where an e-vote system is built from ElGamal).

3.1 The ElGamal Cryptosystem

With specifications due to the previously explained details, ElGamal encryption scheme is defined as follows [ElG85].

ElGamal

Setup: Let p and q be two large primes so that q divides $(p-1)$. Let $\mathbb{G}$ be the subgroup of $\mathbb{Z}_p^\star$ of order q, and g be a generator of $\mathbb{G}$. Let Ω be an (bijective) encoding map from $\mathbb{Z}_q$ onto $\mathbb{G}$.

Key generation: The private key is $x \leftarrow \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Encryption: To encrypt a message $m \in \mathbb{Z}_q$, one encodes m by computing $\omega = \Omega(m)$, randomly selects $r \leftarrow \mathbb{Z}_q$ and computes $(u, v) = (g^r, y^r \cdot \omega)$. The ciphertext is $c = (u, v)$.

Decryption: To decrypt a ciphertext $c = (u, v)$, one computes $\omega = v \cdot u^{-x}$ and recovers the original plaintext $m = \Omega^{-1}(\omega)$, if ω is in the set of all the possible encodings. On the contrary, if ω is not a valid encoding, the decryption process returns a special symbol $\perp$.

This cryptosystem is known to be one-way under the CDH assumption, and indistinguishability holds under the DDH assumption. These security notions are reached in the context of chosen-plaintext attacks, in the standard model.

3.2 The Hash-ElGamal Cryptosystem

In order to overcome the issue of group encoding, a hash variant of ElGamal encryption was suggested.

Hash-ElGamal

Setup: Let p and q be two large primes so that q divides $(p-1)$. Let $\mathbb{G}$ be the subgroup of order q of $\mathbb{Z}_p^\star$, and g be a generator of $\mathbb{G}$. Let $\mathcal{H} : \mathbb{G} \to \{0,1\}^{\ell_m}$ be a hash function.

Key generation: The private key is again $x \leftarrow \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Encryption: To encrypt a message $m \in \{0,1\}^{\ell_m}$, one randomly selects $r \leftarrow \mathbb{Z}_q$ and computes $(u, v) = (g^r, \mathcal{H}(y^r) \oplus m)$. The ciphertext is $c = (u, v)$.

Decryption: To decrypt a ciphertext $c = (u, v)$, one computes $m = \mathcal{H}(u^x) \oplus v$.

This cryptosystem features one-wayness and indistinguishability under chosen plaintext-attacks under the sole CDH assumption. The security proof, however, stands in the random oracle model [FS86, BR93]. Alternatively, under the DDH assumption, one can apply a randomness extractor in place of the random oracle, in order to generate a truly random mask. Unfortunately, this is much less efficient [CFGP06].

Note however that the use of the hash function destroys the homomorphic property of the scheme.

3.3 Encoding-Free ElGamal Encryption

In 2006, Chevallier-Mames, Paillier and Pointcheval proposed an ElGamal variant, using a new encoding-free technique [CPP06]. Their cryptosystem enjoys performances similar to plain ElGamal but does not require group encoding, nor randomness extractors. Furthermore, the security holds in the standard model, but under new intractability assumptions (namely, *Class Diffie-Hellman problems*), the computational of which is shown to be equivalent to CDH problem.

More precisely, their scheme uses the so-called class of an element of subgroup of $\mathbb{Z}_{p^2}$ of order pq, where p and q are two large primes, so that q divides $p-1$. This class is defined as follows. Let $\mathcal{L}$ be described as $\mathcal{L}(w) = (w^q - 1 \bmod p^2)/p$. Let g be a generator of the subgroup of order q of $\mathbb{Z}_p^*$. The class of $w \in \mathbb{Z}_{p^2}$ of

order pq is by definition $[\![w]\!] = \mathcal{L}(w) \cdot \mathcal{L}(g)^{-1} \bmod p$. The encoding-free ElGamal encryption (in its additive variant) is then described as follows.

Encoding-free ElGamal

Setup: Let p and q be two large primes, so that q divides $p-1$. Let g be a generator of the subgroup of order q of $\mathbb{Z}_p^*$.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x \bmod p$.

Encryption: To encrypt a message $m \in \mathbb{Z}_q$, one picks a random $r \in \mathbb{Z}_q$ and computes $u = g^r \bmod p$ and $v = [\![y^r \bmod p]\!] + m \bmod p$. The ciphertext is $c = (u, v)$.

Decryption: To decrypt a ciphertext $c = (u, v)$, one simply computes $m = v - [\![u^x \bmod p]\!] \bmod p$.

We refer to the original paper to show that the security in the standard model under chosen-plaintext attacks is based on the CDH assumption for one-wayness, and on the assumption that the so-called Decision Class Diffie-Hellman is hard for indistinguishability.

Note that this scheme offers a very-weak kind of malleability, in the sense that one can add a plaintext to a ciphertext without decrypting.

4 Main Scheme

This section is the core of our paper. Our goal is to propose a new variant of ElGamal that enjoys both useful malleability properties and a security proof in the standard model (under chosen-plaintext attacks), relatively to a well known assumption. A way to do that is to keep intact the construction of ElGamal and to design a new message encoding that allows some malleability.

First, we remind the Naccache-Stern encryption scheme, whose construction inspired our encoding. Second, we describe our new message encoding for the ElGamal cryptosystem and detail its features. Third, we do exhibit arguments of security for the scheme derived from the combination of ElGamal and our message encoding, and we finally conclude by showing its interesting malleability properties.

4.1 The Naccache-Stern Cryptosystem

The Naccache-Stern cryptosystem is very special in the world of asymmetric cryptography. Indeed, it uses a special trapdoor (a kind of multiplicative knapsack) for deciphering, which makes it unique. The Naccache-Stern scheme [NS97][1] can be described as follows.

[1] We refer the reader to the original paper [NS97] for details on how ℓ_i and p_i are set, in order to achieve optimal efficiency.

Naccache-Stern

Key generation: Let p be a strong prime. For a parameter n, the key generation algorithm searches for n primes p_i and n valuations ℓ_i, such that $\prod_{i=1}^{n} {p_i}^{\ell_i - 1} < p$.

The private key is a random number $x \in \mathbb{Z}_{p-1}^*$. The corresponding public key is the set of elements c_i's defined as $c_i = {p_i}^{1/x \bmod (p-1)} \bmod p$. Then, one defines the set $\mathsf{NS} \in \mathbb{Z}_p^*$ as $\mathsf{NS} = \{\prod_{i=1}^{n} {p_i}^{m_i}, \text{for } m_i \in [0, \ell_i - 1]\}$. For security, the p_i's and NS might be kept private.

Encryption: To encrypt a message $m = \{m_i\}$ with $m_i \in [0, \ell_i - 1]$, one simply computes $w = \prod_{i=1}^{n} {c_i}^{m_i} \bmod p$. The ciphertext is w. Of course, to achieve indistinguishability, some randomization is included in a pre-step (for example, via a padding of the message with random).

Decryption: To decrypt a ciphertext w, one computes $t = w^x \bmod p$. Then, if $t \in \mathsf{NS}$, one simply recovers the message $m = \{m_i\}$ by decomposing t into the base of primes p_i, which is simple as p_i's are small and known. On the contrary, if $t \notin \mathsf{NS}$, the decryption returns a special symbol $\perp$.

Even if the Naccache-Stern scheme is based on the classical DL problem, its special type of trapdoor makes that there does not exist real proof of security for this scheme.

4.2 Our Scheme

As previously said, our scheme is made of the (almost natural but never proposed at our knowledge) composition of ElGamal and a message encoding inspired by the Naccache-Stern cryptosystem (in its optimal bandwidth variant[2]).

Main Points of Design. First, as we want to mix Naccache-Stern and ElGamal schemes, we have to include the p_i's in the subgroup $\mathbb{G} \subset \mathbb{Z}_p^*$ of order q. Unfortunately, taking (relatively) small subgroup order is in contradiction with expecting large bandwidth, as the smaller is q, the more negligible is the probability that a small prime is in $\mathbb{G}$. Thus, we have to take a maximal order for q, *i.e.*, we have to use a strong prime p.

Second, once we have mixed the ElGamal and Naccache-Stern cryptosystems (see below for further details), we see it is no more needed to scramble elements of NS (which will be used to encode the messages in $\mathbb{G}$), as done in Naccache-Stern scheme. Therefore, we need no more to compute and publish large public key set $\{c_i\}$, but rather to take common parameters p_i for all users.

Description. We describe our proposal for a new encoding for the standard ElGamal cryptosystem (see Subsection 3.1) according to the previously explained points.

[2] If, however, the reader prefers to first look at our description with the simplest variant of Naccache-Stern scheme, it suffices to suppose that $\ell_i = 2$ for all i.

Our new encoding

Setup: Let $p = 2q + 1$ be a strong prime. Let g be a generator of the subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$ of order q. For a certain parameter n, the setup algorithm searches for n primes p_i and n valuations ℓ_i, so that p_i's are of order q into $\mathbb{Z}_p^*$ and such that $\prod_{i=1}^{n} {p_i}^{\ell_i - 1} < p$.

Encoding $\Omega(m)$ of a message m: A message m is a $n-$tuple of integers, $m = (m_1, \ldots, m_n)$ such that $0 \leq m_i < l_i$ for all $i = 1, \ldots, n$. The encoding of m is $\Omega(m) = \prod_{i=1}^{n} {p_i}^{m_i}$. We denote NS the image of Ω, *i. e.*, the set of all the possible encodings.

Decoding of an element t of $\mathsf{NS} \subset \mathbb{G}$: To decode, one decomposes t into the base of primes p_i, $t = \prod_{i=1}^{n} {p_i}^{m_i}$ and outputs $m = (m_1, \ldots, m_n)$.

Efficiency. On the one hand, the encoding process is composed of at most n multiplications and n small exponentiations in $\mathbb{Z}$; on the other hand, the decryption is made of a factorization of a simple instance, which is done in practise by some trial divisions by the p_i's. Hence, in term of speed, the cost of the encoding process is negligible compared to the cost of the ElGamal encryption and decryption steps.

Parameter Size. In the Naccache-Stern framework, the public key is made of some secret powers of small primes; our scheme also needs a large set of elements of $\mathbb{Z}_p^*$ (namely, the p_i's), but a definitive advantage over the Naccache-Stern scheme is that (i) these elements can be shared by users instead of being dedicated to a given person, and (ii) these elements are small[3], while c_i's in Naccache-Stern are full-size elements of $\mathbb{Z}_p^*$. These two advantages allow much more practical public-key infrastructures, almost as efficient as those of others ElGamal schemes.

The Impact of the Maximal Order. However, to be fair, one disadvantage of our construction over others ElGamal-based scheme is that we limit the order of the subgroup (that is q) to be maximal, which makes both private key larger and exponentiation longer[4]. We however may consider this as a price to pay to achieve a Naccache-Stern type construction with a provable security.

Encryption Bandwidth. By experimentation, we can estimate the size of the messages that we could encrypt. Typically, using $\ell_i = 2$ for every i, we may be

[3] Typically, one finds the p_i's in the $2n$ first primes, the factor 2 coming from the condition that the chosen primes must be of order q in $\mathbb{Z}_p^*$.

[4] Indeed, the order has been chosen maximal in order to make that almost half small primes are of order q in $\mathbb{Z}_p^*$. However, if the speed is more important than the scheme bandwidth, it should be possible to take shorter orders q at the price of larger p_i's and so less bandwidth.

able to generate p of 1024 bits and p_i's, such that $n \approx 117$, meaning that we would encrypt message of almost 117 bits. For more general case, the point is to optimize $\prod_i \ell_i$ (whose logarithm is the bitsize of messages one can encrypt), under the condition that $\sum_i (\ell_i - 1)\log_2(p_i)$ is limited by the bitsize of p.

Variant. To work in the subgroup of order q, *i. e.*, the subgroup of squares of $\mathbb{Z}_p^*$, we can use another trick instead of using p_i's of order q. As p is a strong prime, $p \equiv 3 \pmod 4$, so given an element x of $\mathbb{Z}_p^*$, either x or $-x$ is a square. As a consequence, in the setup of our system, we can use all small primes, relaxing the condition that p_i must be a square, *i. e.*, of order q. However, we restrict the l_i's such that $\prod_{i=1}^n {p_i}^{\ell_i - 1} < p/2$. In the encoding process, given a message $m = (m_1, \ldots, m_n)$ with $0 \leq m_i < l_i$, we compute $w' = \prod_{i=1}^n {p_i}^{m_i}$ and set $w = w'$ (resp. $w = p - w'$), according to w is a square (resp. a non-square). To decode t, one factors t (resp. $p - t$) if $t < p/2$ (resp. if $t > p/2$).

This variant reduces the public key size (instead of giving all the p_i's we can make n public and the p_i's will be the n first primes), and gives a better bandwidth (if we use $\ell_i = 2$ for every i and if p is a 1024 bit prime, we can encrypt a message of 131 bits as the product of the first 131 primes is smaller than 2^{1022}). The encoding cost is similar to the one of the main scheme (to see efficiently if w is a square, one can pre-compute the Legendre symbols of the p_i's). The decoding process has the same complexity.

Security. We finish this comparison by a major advantage of our scheme (notably over the encoding-free ElGamal scheme based on Class Diffie-Hellman problems): the security of our scheme is based on a classical assumption, namely DDH, as shown in the next section.

4.3 Security Analysis

Oppositely to the typical encryption proofs, we start with indistinguishability, as it appears surprisingly simpler than the analysis of the one-wayness of our scheme.

Theorem 1. *The composition of our new encoding and the* ElGamal *scheme is semantically secure against chosen plaintext attacks, under the* DDH *assumption.*

This theorem follows from the semantic security of the standard ElGamal scheme as we only specify the encoding process of messages in elements of $\mathbb{G}$. □

We emphasize that this semantic security is interesting not only for our scheme, but also as our modification can be seen as a way to achieve a certain security for a Naccache-Stern type cryptosystem (while the original Naccache-Stern scheme has no known proof of security).

One-Wayness. The one-wayness of the construction is not as simple to characterize. At least, due to relations with semantic security, we know it is at least as difficult as DDH to invert the scheme. In addition, one might have the intuition that the scheme is as hard as the CDH to solve, but in the following, we *almost* infirm this, by showing that the natural reduction one could think about is inefficient.

Let a CDH challenge $(g, s = g^a \bmod p, g^x \bmod p)$ on a group $\mathbb{G} = \langle g \rangle$ of order $q = (p-1)/2$ in $\mathbb{Z}_p^*$, described by (g, p, q), and assume an access to an attacker $\mathcal{A}$ against the one-wayness of the scheme corresponding to the composition of our new encoding and ElGamal.

One could give (s, r) as a ciphertext to the attacker (for a random $r \in \mathbb{G}$), which would return a plaintext m if $(rg^{-ax} \bmod p) \in \mathsf{NS}$. However, very few elements of $\mathbb{G}$ are in NS[5], and so the probability of this reduction is very small. Anyway, if such m was returned, one could simply re-encode the returned m by computing $\prod_{i=1}^{n} {p_i}^{m_i} \bmod p$, and divides r by this quantity, in order to get g^{ax}, the answer of the given challenge.

As a conclusion, we only claim that the OW-CPA security of the scheme is at least as difficult as the DDH problem to solve (thanks to the indistinguishability proof), even it might be possible one could find a better proof, based on a weaker assumption.

4.4 Towards a DL-Based Homomorphic Scheme

As we said earlier, additively homomorphic encryption primitives are wanted objects, as they have many applications in protocols. However, today, only schemes based on factorization are known (*e. g.*, Paillier [Pai99], or its predecessors such as Goldwasser-Micali [GM84] and Okamoto-Uchiyama [OU98]).

Remarkably, the encoding-free ElGamal variant of Chevallier-Mames, Paillier and Pointcheval offers a weak kind of malleability, for the first time for a DL-based primitive. As we explain in this section, our scheme goes further, as it allows full *self-blinding* of ciphertexts and a certain malleability on ciphertexts.

Self-blinding. The self-blinding property is very useful, as it allows to re-randomize ciphertexts, which is a key feature for certain applications. In our scheme, as we keep intact the structure of ElGamal, we can still re-randomize a ciphertext (u, v) by picking a random $r \in \mathbb{Z}_q$, then forming the new ciphertext (u', v') of the same plaintext, with $u' = u \cdot g^r \bmod p$ and $v' = v \cdot y^r \bmod p$.

Adding Ciphertexts (Under Restrictions). Let two valid ciphers (u, v) and (u', v'), ciphering respectively two messages $m = (m_1, \ldots, m_n)$ and $m' = (m'_1, \ldots, m'_n)$. Then, $(u'' = u \cdot u' \bmod p, v'' = v \cdot v' \bmod p)$, is a valid cipher of $m'' = (m_1 + m'_1, \ldots, m_n + m'_n)$ as long as one has

$$\forall\, i \in \mathbb{N},\ \ 1 \leq i \leq n,\ \ m_i + {m'}_i < \ell_i. \qquad (\dagger)$$

[5] In fact, $\prod_{i=1}^{i=n} \ell_i$ over the q elements of $\mathbb{G}$.

This means that it is possible to add ciphertexts, if plaintexts were previously selected in such way the condition (†) is always fulfilled.

For example, one might set a voting scheme sketched as follows. The voters would either vote for *yes* (1) or *no* (0) to some terrible question. During the setting of the election, each voter would be assigned an index i she must use (that is, for any $1 \leq i \leq n$, ℓ_i voters would use the same p_i to encrypt their vote $m_i \in \{0, 1\}$). Then, all the votes[6] of the $\sum_i \ell_i$ voters could be "aggregated" by multiplying the ciphertexts, then decrypted by some well-know techniques of threshold encryption (in [Ped91], for instance, one can find the description of a robust threshold variant of ElGamal that can be applied to our scheme). This would give at the end the sum of the votes, and so the result of the ballot. In practice, if p is a 1024 bits prime, if $n = 1$ and $p_1 = 2$, we can manage until 1024 voters with our scheme, which is a satisfying number as in a real life scenario, there are around a thousand registered voters by polling stations.

By using the malleability property, we could also design a multi-candidate election system with n candidates and k voters as follows: a voter would vote for the i^{th} candidate, with $1 \leq i \leq n$, by encrypting p_i. By multiplying all the ciphertexts, the election manager would get an encryption of $\prod_i p_i^{k_i}$ where the k_i's are the number of votes for the i^{th} candidate, respectively. With the setting $k < \log_{p_n}(p)$ (where we supposed that p_n is the larger prime) and $\mathsf{NS} = \{\prod_{i=1}^{n} {p_i}^{m_i}$, for m_i so that $\sum_i m_i \leq k\}$, this encryption is valid, *i. e.*, can be decrypted. If $n = 5$ and p is a 4096 bit prime such that $p_5 = 17$ (that is, one finds a strong prime $p = 2q + 1$ such that 5 of the 7 first primes are of order q), one can still have a thousand voters. With the variant of our encoding, we can use all the primes but with the restriction $k < \log_{p_n}(p/2)$. If $n = 7$ and p is a 4096 bit prime, with this variant, we have $p_7 = 17$ (the seventh prime) and can still have a thousand voters.

We do agree that our scheme is still not the panacea for complete additive homomorphy, but at least, we believe that the full self-blinding of the scheme as well as its restricted additive property might be of interest. Clearly, devising a full and complete DL-based additive scheme is still an open problem.

5 Conclusion

In this paper, we have proposed another way to deal with the problem of message encoding, which is often a forgotten drawback and bottleneck of ElGamal encryption type cryptosystems. Our scheme offers several advantages though some disadvantages: notably, our scheme is based on the classical DDH assumption in the standard model (for the chosen-plaintext scenario), while previous solutions were mainly based on new defined problems or random oracles.

We also showed how our scheme possesses some additive homomorphy, which was an open problem for a long time for DL-type primitives. Notably, we have

[6] That would have been proved to be a correct encryption of a message in the valid set.

shown how it allows full self-blinding and a restricted additivity of the ciphertexts.

Open problems let by this work are of several kinds: firstly, we still consider the search of a full additively homomorphic DL-based scheme as interesting and challenging; secondly, we think it might be possible to achieve a better proof of our scheme in the OW-CPA scenario; lastly, it should be possible to adapt our technique to mix Naccache-Stern and Cramer-Shoup encryption schemes, in order (maybe) to obtain a scheme without encoding that would be secure in the standard model against chosen ciphertext attacks.

Acknowledgement. The work described in this paper has been supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

[BFP+01] Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Proc. of PODC' 01 (2001)

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security – ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)

[CFGP06] Chevassut, O., Fouque, P.-A., Gaudry, P., Pointcheval, D.: The twist-augmented technique for key exchange. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 410–426. Springer, Heidelberg (2006)

[CGS97] Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)

[CPP06] Chevallier-Mames, B., Paillier, P., Pointcheval, D.: Encoding-free ElGamal encryption without random oracles. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 91–104. Springer, Heidelberg (2006)

[CS98] Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)

[DH76] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)

[DJ01] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

[ElG85] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)

[FS86] Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 184–186. Springer, Heidelberg (1987)

[GM84] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)

[Jur03] Jurik, M.: Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols. PhD thesis, Aarhus University (2003)

[Kob87] Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48(177), 203–209 (1987)

[Lip05] Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)

[Mil85] Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)

[NS97] Naccache, D., Stern, J.: A new public-key cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. Springer, Heidelberg (1997)

[NSNK06] Nguyen, L., Safavi-Naini, R., Kurosawa, K.: Verifiable shuffles: a formal model and a Paillier-based three-round construction with provable security. Int. J. Inf. Secur. 5(4), 241–255 (2006)

[OU98] Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)

[Pai99] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)

[Ped91] Pedersen, T.P.: Threshold Cryptosystem without a Trusted Party. In: Proc. of Eurocrypt 1991, pp. 522–526 (1991)

[RSA78] Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)

Differential Properties of Elliptic Curves and Blind Signatures

Billy Bob Brumley[1,⋆] and Kaisa Nyberg[1,2]

[1] Helsinki University of Technology,
Laboratory for Theoretical Computer Science,
P.O. Box 5400, FI-02015 TKK, Finland
{billy.brumley,kaisa.nyberg}@tkk.fi
[2] Nokia Research Center, Finland
kaisa.nyberg@nokia.com

Abstract. Differential uniformity is an important property of cryptographic building blocks used in the design of symmetric ciphers. In this paper it is proved that certain canonical mappings on elliptic curves are differentially uniform. The main observation of this paper is that the impersonation attack against the implicit certificate scheme of Ateniese and de Medeiros does not work if a differentially uniform mapping is used in the scheme. This phenomenon is analyzed in the slightly more general context of a partially blind signature scheme, which is a new cryptographic primitive that seems to gain security properties from differentially uniform mappings.

Keywords: Elliptic curves, differential uniformity, blind signatures, implicit public key certificates, key issuing protocols, provable security, digital signature schemes, message recovery.

1 Introduction

Differential uniformity is an important property of mappings used in the design of symmetric cryptographic primitives such as block ciphers to obtain resistance against differential cryptanalysis [1]. Differential uniformity means that for all fixed strict differences in the input there is a large uncertainty about the output difference. Previously, it is known from [2] and [3] that the mapping from the multiplicative group of a finite field to the additive group of the finite field is differentially uniform. This paper contains two new contributions. First, it is proved that the canonical mapping from an elliptic curve to its x-coordinate is differentially uniform. Secondly, a potential application of differential uniformity to public key cryptography is outlined.

Ateniese and de Medeiros [4] presented a modification (mNR) of the Nyberg-Rueppel (NR) signature scheme [5]. They also proposed a scheme for generating implicit public key certificates based on mNR signatures by a trusted third party (TTP). While a solution was presented to the key escrow problem by blinding

⋆ Supported by the project "Packet Level Authentication" funded by TEKES.

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 376–389, 2007.

TTP to the user's private key, the solution requires a proof of knowledge of a discrete logarithm of the blinding value to prevent an impersonation attack.

The main observation given in this paper is that if a differentially uniform mapping is used in the generation of implicit certificates then the impersonation attack does not work. This indicates that the proof of knowledge of the discrete logarithm could be eliminated and the performance of the protocol could be improved. It is anticipated that the solution herein can also be applied to other cryptographic protocols and schemes based on the difficulty of the discrete logarithm problem.

We start by proving the differential uniformity property of elliptic curves. Sec. 2 contains background information on elliptic curves. Elliptic curve point addition differentials with respect to taking x-coordinates are studied in Sec. 3, and differential uniformity properties thereof proved. In Sec. 4 we review the implicit certificate scheme by Ateniese and de Medeiros and the impersonation attack against it. The new partially blind NR signatures are presented in Sec. 5, and a security property based on differential uniformity is proved. We conclude in Sec. 6.

2 Elliptic Curves

In general, elliptic curves over an arbitrary field K are defined by the Weierstrass equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ . \tag{1}$$

An Abelian group can be formed from the points on the curve along with the point $\mathcal{O}$, which serves as the identity element. A line through two points on the curve (or a line tangent to the curve if the two points are the same) intersects the curve at only one other point.

As stated, K can be any field. However, elliptic curves used in cryptography [6,7] are defined over a large prime field $\mathbb{F}_p$ or large binary field $\mathbb{F}_{2^m}$ where the order of an elliptic curve E (denoted $\#E$) is large and nearly prime. This allows for the creation of elliptic curve analogues of common schemes, e.g. ECDSA and DSA [8].

2.1 Elliptic Curves over Prime Fields

Over $\mathbb{F}_p$ with $p > 3$ and prime, every elliptic curve is isomorphic to a curve given by

$$y^2 = x^3 + ax + b. \tag{2}$$

Point Addition and Doubling. The sum of two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ is calculated as follows.

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \text{ where} \end{aligned} \tag{3}$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \dfrac{3{x_1}^2 + a}{2y_1} & \text{if } P = Q \end{cases} \tag{4}$$

Here λ is the slope determined by P and Q.

2.2 Elliptic Curves over Binary Fields

Over $\mathbb{F}_{2^m}$, every elliptic curve[1] is isomorphic to a curve given by

$$y^2 + xy = x^3 + ax^2 + b. \tag{5}$$

Point Addition and Doubling. The sum of two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ is calculated as follows.

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \tag{6}$$

$$y_3 = \lambda(x_1 + x_3) + y_1 + x_3, \text{ where}$$

$$\lambda = \begin{cases} \dfrac{y_2 + y_1}{x_2 + x_1} & \text{if } P \neq Q \\ x_1 + \dfrac{y_1}{x_1} & \text{if } P = Q \end{cases} \tag{7}$$

As above, λ is the slope determined by P and Q.

3 Elliptic Curve Point Addition Differentials

In what follows, it is shown that the mapping which maps a point of an elliptic curve to its x-coordinate is differentially uniform, or formally (as illustrated in Fig. 1)

$$\max_{d \in \mathbb{F}, D \in E(\mathbb{F}) \setminus \{\mathcal{O}\}} \#\{P \in E(\mathbb{F}) \setminus \{\mathcal{O}, -D\} \mid (P + D)_x - P_x = d\}$$

is sufficiently small. Due to the nature of elliptic curves, this definition differs slightly from the one presented more generally later (Sec. 5.1, Def. 2); in particular, the x-coordinate of the identity element $\mathcal{O}$ (point-at-infinity) is not defined.

The two cases of elliptic curves for cryptographic use over prime fields and binary fields are explored. How to use these results in practice will be discussed later in Sec. 5.2.

Theorem 1. *On an elliptic curve E over a prime field $\mathbb{F}_p$, given an arbitrary but fixed point $D \in E(\mathbb{F}_p) \setminus \{\mathcal{O}\}$ and field element $d \in \mathbb{F}_p$, the number of points $P \in E(\mathbb{F}_p) \setminus \{\mathcal{O}, -D\}$ such that $(P + D)_x - P_x = d$ holds is at most 4.*

[1] Only non-supersingular curves are considered here.

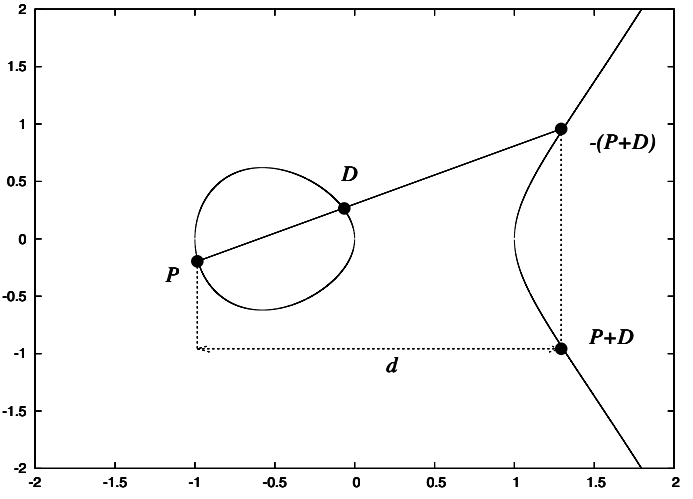


Fig. 1. Elliptic curve point addition differential over $\mathbb{R}$

Proof. Let x, y denote the x and y coordinates of P and u, v of D. Given (3),

$$\left.\begin{aligned}(P+D)_x &= \lambda^2 - x - u\\ (P+D)_x &= d + x\end{aligned}\right\} \longrightarrow 2x = \lambda^2 - u - d \tag{8}$$

where λ is the slope. Clearly $\lambda(x-u) = y - v$ from (4) holds in both the cases of $P \neq D$ and $P = D$, so substituting in (8) yields

$$2y = \lambda(2x - 2u) + 2v = \lambda^3 - 3\lambda u - \lambda d + 2v. \tag{9}$$

As both points P and D are on the curve, they both satisfy (2). The difference of these two equations is

$$\begin{aligned} y^2 - v^2 &= x^3 + ax - u^3 - au, \text{ which factors into}\\ (y-v)(y+v) &= (x-u)(x^2 + xu + u^2 + a), \text{ and given (4)},\\ (y+v)\lambda &= x^2 + xu + u^2 + a. \end{aligned} \tag{10}$$

Clearly this holds for $P \neq D$. If $P = D$, then $x = u$ and $y = v$ and the last equation gives the same expression for λ as (4) and thus it holds for $P = D$ as well. We now eliminate x and y from (10) by substituting in (8) and (9) and simplifying, which yields

$$\lambda^4 - 6u\lambda^2 + 8v\lambda - 3u^2 - d^2 - 4a = 0.$$

As this equation has degree four, there are at most four solutions for λ. Moreover, $P = (x, y)$ is uniquely determined given $\lambda, D = (u, v), d$. Specifically, x is calculated from (8) given λ, d, u; y is calculated from (9) given λ, u, v, d. Thus, there are at most four different points P which satisfy this equation. □

Theorem 2. *On an elliptic curve E over a binary field $\mathbb{F}_{2^m}$, given an arbitrary but fixed point $D \in E(\mathbb{F}_{2^m}) \setminus \{\mathcal{O}\}$ and field element $d \in \mathbb{F}_{2^m}$, the number of points $P \in E(\mathbb{F}_{2^m}) \setminus \{\mathcal{O}, -D\}$ such that $(P+D)_x - P_x = d$ holds is at most 4.*

Proof. The same notation as above is used. Given (6),

$$\left.\begin{aligned}(P+D)_x &= \lambda^2 + \lambda + x + u + a \\ (P+D)_x &= d + x\end{aligned}\right\} \longrightarrow \lambda^2 + \lambda + u + a + d = 0$$

As this is a quadratic equation, there are either two or zero solutions for the slope λ, and therefore at most two lines passing through D with these given slopes. A line through D intersects the curve in at most two other points. This gives a total of at most four points. □

4 Implicit Certificates and Impersonation Attacks

We use multiplicative group notation throughout this and the following section unless specified otherwise. The group is denoted by $\langle G \rangle$ where G is the generator element of the group. The order of the group is denoted by q, and in our application it is typically selected to be a prime. The digital signature schemes discussed in this paper are based on the difficulty of the discrete logarithm problem in $\langle G \rangle$. Given a private key $v \in \mathbb{Z}_q$ the public key is computed as $V = G^v$. The digital signature schemes also make use of a hash function H, which maps strings of arbitrary length to $\mathbb{Z}_q$ and is assumed to be secure under preimage, second preimage and collision attacks. The signatures also make use of a projection P (or P_1), which maps elements of $\langle G \rangle$ to $\mathbb{Z}_q$. The security properties of the projection are in the main focus of this paper.

4.1 Modified NR Signature Scheme

In [4] Ateniese and de Medeiros presented a modification (mNR) of the Nyberg-Rueppel (NR) signature scheme. In addition to G and q, the system parameters include a group element G_1 such that the discrete logarithm of G_1 to the base G is known neither to the signer nor to potential forgers of signatures. To sign a message m with hash code $h = \mathrm{H}(m)$, the signer takes a random $k \in \mathbb{Z}_q$, computes $R = G^k G_1^h$, $r = \mathrm{P}_1(R)$, and $s = -k - vr \pmod q$. The signed message is (m, R, s). The verification of a signed message (m, R, s) is based on checking the equality $G_1^h = RG^s V^r$.

In [4], two types of implementations were considered where the finite group $\langle G \rangle$ can be either the multiplicative group of a finite field or the group of K-rational points on an elliptic curve over a finite field K. In the former case the projection P_1 is the canonical embedding of the multiplicative group to the field. In the latter case, the projection P_1 first computes the compression of the point $R = G^k G_1^h$ to obtain one bit for the y-coordinate and a field element as the x-coordinate. The x-coordinate is then considered as a bit-string to which

the y-coordinate bit is prepended. This string is then converted to an integer, which is then reduced modulo q.

While a security proof is given for the mNR scheme, its application to implicit certificates implies new security requirements that are not covered by the security properties of the mNR scheme. The problem lies in the fact that when generating an implicit certificate, the TTP actually generates a signature on a message that consists of two parts: the public key of the user $U = G^u$ submitted by the user with identity ID, and the identity ID. The ID part is handled as the message m in the mNR scheme, but the public key is integrated in the first part R of the signature to be recovered in the verification of the signature, similarly as messages are recovered in the original NR scheme.

As suggested in [4] by submitting a suitable U to the TTP a malicious user can forge an implicit certificate on any identity and on a public key selected by the user. The implicit certificate scheme based on the mNR scheme is discussed next in more detail.

4.2 Implicit Certificates

As an application of the mNR signature scheme, a public key issuing protocol with implicit certificates was presented in [4]. Implicit certificates are comprised of a user's identity and some reconstruction public data, which together with the TTP's public key is used to reconstruct the user's public key [9].

To prevent key escrow of the private keys, a blinding value is used. For the implicit certificate to be useful to the submitter of the public key request, the discrete logarithm of the blinding value must be known by the submitter.

Let sig_u and ver_U denote the algorithms for signature generation with private key u and verification with public key U of a signature scheme where the generated keys are intended to be used. Given an identity ID_A of Alice, the following protocol is executed between Alice and the public key issuer TTP with private key v and public key V. The hash function $\mathrm{H}_1(\cdot)$ used below is short notation for $G_1^{\mathrm{H}(\cdot)}$.

$$\begin{aligned}
&\text{TTP} \longleftarrow \text{Alice: } G^{k_A}\\
&\text{Alice} \longleftarrow \text{TTP: } C\\
&\text{TTP} \longleftarrow \text{Alice: } sig_{k_A}(C)\\
&\qquad\quad \text{TTP: } ver_{G^{k_A}}(sig_{k_A}(C))\\
&\text{Alice} \longleftarrow \text{TTP: } \begin{cases} R_A = & G^{k_A} G^{k\,\mathrm{H}_1(ID_A)} \\ \overline{s}_A = & -k - v\,\mathrm{P}_1(R_A) \pmod q \end{cases}
\end{aligned} \tag{11}$$

Alice's private key is $s_A = \overline{s}_A - k_A \pmod q$. The implicit certificate is R_A. Given the triple $\{R_A, ID_A, V\}$, Alice's public key G^{s_A} extracts correctly:

$$\frac{\mathrm{H}_1(ID_A)}{V^{\mathrm{P}_1(R_A)} R_A} = \frac{\mathrm{H}_1(ID_A)}{G^{v\mathrm{P}_1(R_A)} G^{k_A} G^{k\,\mathrm{H}_1(ID_A)}} = G^{k+\overline{s}_A-\overline{s}_A+s_A-k} = G^{s_A}$$

The second and third messages of the protocol are exchanged because Alice must prove her knowledge of the secret exponent k_A. For this purpose, TTP issues a challenge message C. Alice then signs this message using key k_A and TTP verifies this signature using public key G^{k_A} (the element that Alice submitted).

The proof of possession is done in order to prevent a straightforward impersonation attack which goes as follows. Instead of sending an exponential with a known exponent, the attacker selects a suitable value that allows forgery of the identity. Consider a malicious, but legitimate user Malice attempting to obtain a valid signature from TTP on Alice's identity using (11) where no proof of knowledge is performed. To succeed, Malice, with identity ID_M, chooses a difference[2] D such that the value of $\mathrm{P}_1(R_A)$ which represents the data to be signed by the TTP is not changed if the identity is changed. This holds if

$$DG^{k_M}G^{k}\mathrm{H}(ID_M) = G^{k_M}G^{k}\mathrm{H}(ID_A) \text{ , that is,}$$
$$D = \frac{\mathrm{H}_1(ID_A)}{\mathrm{H}_1(ID_M)}. \tag{12}$$

Thus, by submitting DG^{k_M} instead of G^{k_M} Malice obtains a public key on Alice's identity with a private key known to Malice. Hence in both implementations, either a multiplicative group of a finite field or an elliptic curve, an attacker can compute the difference D for any selected target identity ID_A.

4.3 Thwarting Impersonation Attacks

Let us consider a slight modification to the implicit certificates discussed above. Instead of (11) let the output from TTP to Alice be computed as follows:

$$\text{Alice} \longleftarrow \text{TTP: } \begin{cases} R_A = & G^{k_A}G^{k} \\ \overline{s}_A = & -k - v(\mathrm{P}_1(R_A) + \mathrm{H}_1(ID_A)) \pmod q. \end{cases} \tag{13}$$

If Malice wants to launch a similar impersonation attack against this scheme, he should find a difference D such that by submitting DG^{k_M} instead of G^{k_M} he has

$$\mathrm{P}_1(DG^{k_M}G^{k}) + \mathrm{H}(ID_M) = \mathrm{P}_1(G^{k_M}G^{k}) + \mathrm{H}(ID_A), \text{ that is,}$$
$$\mathrm{P}_1(DG^{k_M}G^{k}) - \mathrm{P}_1(G^{k_M}G^{k}) = \mathrm{H}(ID_A) - \mathrm{H}(ID_M). \tag{14}$$

Here G^k is a random group element, which the TTP generates after Malice has submitted its blinding value DG^{k_M}. The main result of this paper is to show that if the projection P_1 is differentially uniform, then it is unlikely that Malice will succeed.

Another solution to prevent impersonation attacks previously presented in the literature in the context of signatures providing partial message recovery [10,11],

[2] The term *difference* in this sense is as used in differential cryptanalysis and also for quotients in multiplicative groups.

is to use a hash function. This solution is outlined in Appendix A. But in order to prove that the impersonation attack does not work, non-standard assumptions of the hash function may be required. Another drawback of the hash-function based solution is that it slows down signature generation, as the hash function must be computed online.

In the next section we analyze our solution in more detail in the context of partially blind signatures. Our solution allows a concrete security proof and does not require online computation of a hash function.

5 Partially Blind Signatures

Partially blind signatures to be discussed next have applications wherever the message is split into two parts: a blinded part and a part known to the signer. The purpose of such a signature is to prove that the signer approves the known part of the message as valid and provide a binding between the blinded and known parts of the message. A typical example would be a time-stamping application where data in encrypted form is submitted at a certain time to be time-stamped. Thus, a binding between the time and the data is provided by a blind signature. In this paper the application is the implicit certificate scheme, where the signer is TTP and a binding between the identity of the user and the user's public key is provided without seeing the private key.

Usually only a small proportion of all possible blinded data is useful to the user. The threat model is that a malicious user can submit as blinded data something that makes it possible to apply the binding created by the signer to different data that is more useful to the user. For example, it is important to guarantee that the attacker cannot, by submitting garbage at a certain time, obtain a binding between a time-stamp (with a different time on it) and some meaningful data.

Undoubtedly, signature schemes realizing such bindings between different message parts is an interesting cryptographic primitive and would deserve a formal general treatment. However, for the purposes of this paper we restrict ourselves to the NR signature scheme giving partial message recovery, which is used by the TTP to generate the signatures in (11) and (13).

To describe this signature scheme we use the same notation as in Sec. 4. To sign the message (m, B), where $B \in \langle G \rangle$ represents the blinded part of the message, the signer takes a random $k \in \mathbb{Z}_q$, computes

$$\begin{aligned} R &= G^k B\,, \\ r &= f(R, m) \text{ and} \\ s &= -k - vr \pmod{q}. \end{aligned} \tag{15}$$

The signed message is (m, R, s). The function f maps group elements R and messages m to integers in $\mathbb{Z}_q$. In the verification B is recovered from (m, R, s) by computing $r = f(m, R)$ and using

$$G^s V^r R = G^{s+rv+k} B = B.$$

It is not essential to restrict to NR signatures; similar partially blind signatures can be obtained using the PECDSA scheme [11] with message recovery as shown in Appendix B.

5.1 Security of Partially Blind NR Signatures

The partially blind signatures are used as follows: a user creates a message (m, B), where m is the known part and B is the blinded part of the message, and submits them to the signer who generates the signature (R, s).

For the same reason why digital signatures giving message recovery are vulnerable to existential attacks, it is clear that partially blind signatures are vulnerable to existential attacks. Furthermore, it is easy to see that for any given messages m and $\hat{m}$ it is possible to find B and signatures (R, s) of (m, B) and $(\hat{R}, \hat{s})$ of $(\hat{m}, B)$ by submitting only one of the messages to the signer. However, if the blinded parts B are restricted to a subset $\mathcal{B}$ of $\langle G \rangle$, one can obtain a reasonable and useful security definition. The security of partially blind signatures is considered with respect to this subset $\mathcal{B}$. In the application of implicit certificates, the set $\mathcal{B}$ is the set of group elements for which a user knows the discrete logarithm. In the time stamping application, $\mathcal{B}$ consists of the encryptions of meaningful data to be time-stamped. Hence we allow that given $B \in \langle G \rangle$, the signer is not able to tell whether $B \in \mathcal{B}$ or not. Next we state the main security requirement of partially blind signatures.

Definition 1. *Partially blind NR signatures with function f and set $\mathcal{B}$ are said to be secure against message forgery if it is infeasible to find $B \in \mathcal{B}$, messages m and $\hat{m}$, $m \neq \hat{m}$, and a group element $\hat{B} \in \langle G \rangle$ such that by submitting $(\hat{m}, \hat{B})$ to the signer and obtaining a valid signature to it, the user obtains a valid signature for (m, B).*

The security definition requires that the known part m cannot be forged. Note that changing the blinded part B is easy: if (R, s) is a signature on a message (m, B), then $(R, s + e \bmod q)$ is a signature on $(m, G^e B)$.

On the other hand, if we had a partially blind NR signature scheme that satisfies Definition 1 with $\mathcal{B}$ being the set of group elements for which a user knows the discrete logarithm, we would have a scheme for implicit certificates where the proof of knowledge of the discrete logarithm is not needed to prevent the impersonation attack.

We observe that forgeries violating the security definition can be divided into two categories:

Attack 1. Given a valid signature $(\hat{R}, \hat{s})$ on a message $(\hat{m}, \hat{B})$ possibly chosen by the adversary, generate without knowledge of the private key a valid signature (R, s) for a message (m, B), where $B \in \mathcal{B}$ and $\hat{r} = f(\hat{m}, \hat{R}) \neq f(m, R) = r$.

Attack 2. Given a valid signature $(\hat{R}, \hat{s})$ on a message $(\hat{m}, \hat{B})$ possibly chosen by the adversary, generate without knowledge of the private key a valid signature (R, s) for a message (m, B), where $B \in \mathcal{B}$ and $\hat{r} = f(\hat{m}, \hat{R}) = f(m, R) = r$.

The goal of this paper is to prove resistance against Attack 2 where r is kept unchanged. This is the kind of attack that was discussed in Sec. 4. To justify security against Attack 1 we present only informal arguments. Partially blind signatures can be considered as a combination of two well established signature schemes: the NR scheme giving message recovery, and the NR signature scheme using hash functions. In the message recovery case, the message part m is omitted ($\mathrm{H}(m) = 0$), and in the latter case the recovered part B is omitted ($B = 1$). We believe that resistance against Attack 1 is independent of the way the message input is formatted, and therefore inherited to partially blind signatures from the underlying signature scheme.

On the other hand, resistance against Attack 2 clearly depends on the function f. As shown in Sec. 4.2 the function $f(m, R) = \mathrm{P}_1(R\mathrm{H}_1(m))$ used in generating the signatures by TTP in (11) is vulnerable to Attack 2.

Next we show that the function $f(m, R) = \mathrm{P}(R) + \mathrm{H}(m) (\mathrm{mod} q)$ used in (13) can give resistance against Attack 2 depending on the properties of P. The property we are going to need is differential uniformity defined in [2] as follows.

Definition 2. *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two groups and* P *a function from $\mathcal{G}_1$ to $\mathcal{G}_2$. Let δ be a positive integer. We then say that the function* P *is differentially δ-uniform if for all $D_1 \in \mathcal{G}_1$, $D_1 \neq 1$, and $D_2 \in \mathcal{G}_2$ it holds that*

$$\#\{X \in \mathcal{G}_1 \mid \frac{\mathrm{P}(D_1 X)}{\mathrm{P}(X)} = D_2\} \leq \delta.$$

Theorem 3. *Consider the partially blind NR signature scheme described in (15) with function f defined as $f(m, R) = \mathrm{P}(R) + \mathrm{H}(m) (\mathrm{mod} q)$, where* H *is a secure hash function, and with a set $\mathcal{B}$. Suppose that the projection* P *from $\langle G \rangle$ to $\mathbb{Z}_q$ is differentially δ-uniform and let ω describe the computational bound of the work of the adversary. Then the probability that Attack 2 succeeds and a blind NR signature is forged on one message (m, B), where $B \in \mathcal{B}$, is at most $(\#\mathcal{B} + \delta\omega)/q$.*

Proof. The success probability of an existential attack is upperbounded by $\#\mathcal{B}/q$. Let us now suppose that when running Attack 2, the adversary selects a blinded part B, where $B \in \mathcal{B}$ and generates a message $(\hat{m}, \hat{B})$, where $\hat{B} \in \langle G \rangle$ and submits it to the signer. Then the attacker gets a signature $(\hat{R}, \hat{s})$, where $\hat{R} = G^{\hat{k}}\hat{B}$ and $G^{\hat{k}}$ is a random group element generated by the signer. Suppose now that from here the adversary can compute a valid signature (R, s) on (m, B) for some message $m \neq \hat{m}$. Then in Attack 2 the following holds

$$\mathrm{P}(R) + \mathrm{H}(m) \equiv \mathrm{P}(\hat{R}) + \mathrm{H}(\hat{m}) \, (\mathrm{mod} q). \tag{16}$$

If $s \neq \hat{s}$ we can replace $\hat{B}$ by $\hat{B}G^{s-\hat{s}}$ to get a valid signature $(\hat{R}, s)$ for a message $\hat{B}$. Hence we can assume that $s = \hat{s}$ without loosing generality. As also $r = \hat{r}$, it then follows that

$$\frac{R}{\hat{R}} = \frac{B}{\hat{B}}$$

and hence this quotient is a group element determined by the submitted value of $\hat{B}$. Let us denote

$$D = \frac{B}{\hat{B}} \text{ and } d = \mathrm{H}(\hat{m}) - \mathrm{H}(m) (\mathrm{mod} q),$$

where d is a quantity fixed by the selection of m. Then we get from (16) that

$$\mathrm{P}(\hat{R}D) - \mathrm{P}(\hat{R}) \equiv d \pmod{q} \tag{17}$$

holds for $\hat{R}$. As P is differentially δ-uniform there are at most δ such elements $\hat{R}$ in $\langle G \rangle$. Based on one submitted value $\hat{B}$ the adversary can try only a limited number, say σ, different values of $\hat{B}G^{s-\hat{s}}$ by changing s. This means that the adversary can try to establish (17) for only σ different D. Similarly, by varying m the adversary can try to establish (17) for a number, say μ different values of d. As $\hat{R}$ is a random element in a group of order q the probability of success for each fixed pair D and d is upperbounded by δ/q. The number of different pairs s and m to be tried is upperbounded by $\sigma\mu \leq \omega$. It follows that adversary's success probability for a fixed $B \in \mathcal{B}$ is upperbounded by $\delta\omega/q$. The claimed upperbound follows by taking the probability of the existential attack into account. □

5.2 Differentially Uniform Projections

Let p be a prime and G a generator of the multiplicative group modulo p. Then for all R and D in $\langle G \rangle$ we have $RD - R \equiv R(D-1) \pmod{p}$. For a fixed difference D, $D \neq 1$ this value runs through the entire group as R varies. Hence the canonical embedding from the multiplicative group of $\mathbb{Z}_p$ to the additive group of $\mathbb{Z}_p$ is differentially 1-uniform.

Let ε be the smallest integer such that $p \leq \varepsilon q$. Given $d \in \mathbb{Z}_q$, let x and $y \in \mathbb{Z}_p$ be such that $x - y \equiv d \pmod{q}$. Then $-p < x - y < p$ from where it follows that $x - y$ may take $2\varepsilon - 1$ different values, that is, $x - y = d + iq$, where $i \in \{-\varepsilon + 1, \ldots, -1, 0, 1, \ldots, \varepsilon - 1\}$. This gives at most $2\varepsilon - 1$ values of $x - y$ modulo p. It follows that the mapping from the multiplicative group of $\mathbb{Z}_p$ to the additive group $\mathbb{Z}_q$ is differentially δ-uniform with $\delta = 2\varepsilon - 1$.

In Sec. 3 it was shown that the projection which maps elliptic curve point to its x-coordinate is differentially 4-uniform. For elliptic curves over prime fields (for prime p) it then follows that the projection P is differentially 4δ-uniform with $\delta = 2\varepsilon - 1$.

The canonical mappings from the multiplicative group of a binary field and from an elliptic curve over a binary field to the additive group (with exclusive-or addition) of the field are differentially uniform. The problem arises when the string in the binary field is converted to an integer and then reduced modulo q. In general, a fixed integer difference can be taken by pairs of field elements with a number of different differences with respect to exclusive-or addition. In differential cryptanalysis of block ciphers, a common albeit heuristic assumption is that a composition of two essentially different functions, of which one is differentially uniform, is also differentially uniform. However, what essentially different means

in this context lacks precise definition. Hence we can only conjecture that the projection P is differentially δ-uniform with a small constant δ in the case of a binary field as well. This conjecture is also supported by our experiments.

6 Conclusion

In this paper, we proved that the canonical mapping on an elliptic curve that maps a point to the x-coordinate of the point is differentially uniform. This property has previously been known for mappings that map the elements of the multiplicative groups of a finite field to the additive group of the field. We also proposed a potential application of this property to enhance the security of the implicit certificate scheme by Ateniese and de Medeiros. In particular, we proved that if differential uniform projections are used when computing the signatures then the certificates issued on a certain identity cannot be misused to provide certificates to some other identity. We achieve this property without the use of key escrow. The proof is valid when the implicit certificate scheme is implemented in a multiplicative group of a prime field or on an elliptic curve over a prime field. It remains an open question whether this result holds also in the case of a binary field.

Our results indicate that the property of differential uniformity can be useful for enhancing the security and efficiency of protocols and schemes based on asymmetric cryptography. It would be interesting to find more such applications.

Acknowledgements

We thank Tanja Lange for the proof of Theorem 2. We also thank her and the anonymous referees for insightful comments that helped us to improve the paper.

References

1. Rijmen, V., Daemen, J.: The Design of Rijndael. AES – The Advanced Encryption Standard. Springer, Heidelberg (2002)
2. Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, Springer, Heidelberg (1994)
3. Granboulan, L., Levieil, É., Piret, G.: Pseudorandom permutation families over abelian groups. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 57–77. Springer, Heidelberg (2006)
4. Ateniese, G., de Medeiros, B.: A provably secure Nyberg-Rueppel signature variant with applications. Cryptology ePrint Archive, Report,2004 /093 (2004), http://eprint.iacr.org/2004/093
5. Nyberg, K., Rueppel, R.A.: A new signature scheme based on the DSA giving message recovery. In: CCS '93: Proceedings of the 1st ACM conference on Computer and communications security, pp. 58–61. ACM Press, New York (1993)
6. Miller, V.S.: Use of elliptic curves in cryptography. In: CRYPTO 1985: Advances in Cryptology, pp. 417–426. Springer, London, UK (1986)

7. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
8. FIPS: Digital signature standard (DSS). FIPS PUB 186-2 (+ Change Notice). Technical report, U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology (January 2000)
9. Brown, D., Gallant, R., Vanstone, S.: Provably secure implicit certificate schemes. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 147–156. Springer, Heidelberg (2002)
10. Pintsov, L.A., Vanstone, S.A.: Postal revenue collection in the digital age. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 105–120. Springer, Heidelberg (2001)
11. Granboulan, L.: PECDSA. how to build a dl-based digital signature scheme with the best proven security. NESSIE Technical Report NES/DOC/ENS/WP5/022/1 (2002), http://eprint.iacr.org/2002/172

A Avoiding Proof of Possession Using a Hash Function

The proof of possession of the discrete logarithm of the submitted value can be avoided by providing a stronger binding between the two message parts under the signature. One solution, previously presented in the literature in the context of signatures providing partial message recovery [10], [11], is to use a hash function.

We first recall the provably secure signature scheme PECDSA by Louis Granboulan presented in [11]. PECDSA is intended for implementation on elliptic curves with the projection P that maps the point first on its x-coordinate, and then to $\mathbb{Z}_q$. The scheme makes use of a redundancy function ρ that adds necessary redundancy to the part of the message that is recovered in the verification procedure. For a given message $\bar{m}$, redundancy is added as $\rho(\bar{m}) = (\bar{m}||0...0)$, where the number of zeroes is determined by the security parameter.

To sign the message $(\hat{m}, \bar{m})$, where $\bar{m}$ is the recoverable part in $\mathbb{Z}_q$, the signer takes a random $k \in \mathbb{Z}_q$, computes $R = G^k$, $r = \rho(\bar{m}) \oplus \mathrm{P}(R)$, $h = \mathrm{H}(\hat{m}, r)$ and $s = v^{-1}(k - (h \oplus r)) \pmod q$. The signed message is $(\hat{m}, r, s)$. The verification of $(\hat{m}, r, s)$ begins with computation of $h = \mathrm{H}(\hat{m}, r)$. Then R is recovered as $R = G^{h \oplus r} V^s$. Finally, the verifier checks that the redundancy of the message $\bar{m}$ recovered from $r \oplus \mathrm{P}(R)$ satisfies $\rho(\bar{m}) = r \oplus \mathrm{P}(R)$. Then PECDSA is provably secure against existential forgery assuming that $\langle G \rangle$, H and P are idealized in a manner specified in [11].

Using a hash function as above, we can replace (11) with the following.

$$\begin{aligned} \text{TTP:}\quad R_A &= G^{k_A} G^k \\ r &= \mathrm{H}(R_A, ID_A) \oplus \mathrm{P}(R_A) \\ \overline{s}_A &= -k - vr \pmod q \\ \text{Alice} \longleftarrow \text{TTP:}\quad & \{R_A, \overline{s}_A\} \end{aligned} \tag{18}$$

Alice's private key is $s_A = \overline{s}_A - k_A$ as before. The implicit certificate is R_A. Given the triple $\{R_A, ID_A, V\}$, Alice's public key G^{s_A} is now extracted by computing r from ID_A and R_A and then computing $V^{-r} R_A^{-1} = G^{s_A}$.

The attack described in Section 4.2 does not seem to work anymore. But in order to prove that it does not work, non-standard assumptions on the hash function may be required. Another drawback of the hash-function based solution is that it slows down signature generation, as the hash function must be computed online.

B Partially Blind Signatures Based on PECDSA

Let us discuss the following modification of PECDSA. Given a message (m, B) the signature is (R, s) where

$$\begin{aligned} R &= G^k B \\ r &= \mathrm{H}(m) + \mathrm{P}(R) \pmod{q} \\ s &= v^{-1}(-k - r) \pmod{q}. \end{aligned}$$

If the projection P is differentially uniform then secure partially blind signatures can be obtained. To achieve this we have removed the value R from the inputs of the hash function H of the original PECDSA scheme. By doing this, the provable security may be lost. It is an open problem if there is a provable secure signature scheme giving partial message recovery where the recoverable part is blinded and Attack 2 is infeasible.

Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication Using Multibase Number Representation

Pradeep Kumar Mishra and Vassil Dimitrov

Centre for Information Security and Cryptography University of Calgary
Calgary, Canada
pmishra@ucalgary.ca, dimitrov@atips.ca

Abstract. In the current work we propose two efficient formulas for computing the 5-fold ($5P$) of an elliptic curve point P. One formula is for curves over finite fields of even characteristic and the other is for curves over prime fields. Double base number systems (DBNS) have been gainfully exploited to compute scalar multiplication efficiently in ECC. Using the proposed point quintupling formulas one can use 2, 5 and 3, 5 (besides 2, 3) as bases of the double base number system. In the current work we propose a scalar multiplication algorithm, which uses a representation of the scalar using three bases 2, 3 and 5 and computes the scalar multiplication very efficiently. The proposed scheme is faster than all sequential scalar multiplication algorithms reported in literature.

Keywords: Elliptic Curve Cryptosystems, Scalar Multiplication, Quintupling, Efficient Curve Arithmetic.

1 Introduction

Undoubtedly, the papers [25,28], which independently proposed elliptic curve cryptography (ECC), are among the most cited papers in cryptology. In ECC, elliptic curves over finite fields are used to generate finite abelian groups to implement public key cryptographic primitives. The advantage of using elliptic curve groups is: there is no known subexponential algorithm to solve the elliptic curve discrete logarithm problem (ECDLP). This means that a desired security level can be achieved with a much smaller key size in comparison to other public key schemes. This, in turn, leads to efficient implementation and efficient use of transmission bandwidth. Another advantage of ECC is the flexibility it offers in the choice of various security parameters (like group order and representation of its elements, group arithmetic, underlying field and its representation etc) used in its implementation.

Scalar multiplication of elliptic curve points is one of the most researched operations in cryptography. If P is a point on an EC and d is an integer, the

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 390–406, 2007.

operation computing the d-fold of P, namely the point dP, is called scalar multiplication. Several methods have been reported in literature to compute scalar multiplication efficiently and securely from prying eyes (side-channel attackers). The strategies used for enhancement of efficiency are: (1) efficient group arithmetic in the elliptic curve group, (2) using a "nice" representation for the scalar (the sparser, the better), (3) use of precomputation to precompute some points required later (4) using efficient algorithms like the sliding window method, comb methods or use of efficient addition chains, like Montgomery's ladder etc.

In the current work, we propose a new scalar multiplication algorithm, the essence of whose efficiency comes from two new efficient point quintupling formulas for curves over arbitrary prime and binary fields and use of a very sparse representation of the scalar using three bases. For the last couple of years, double base number systems (DBNS) have been proposed for use in this context by several authors [1,2,8,12,13,16]. For general curves, a DBNS representation of the scalar using 2 and 3 as bases has been proved quite efficient [12]. In search of sublinear scalar multiplication algorithms, authors of [1] have used complex bases, 3 and τ for Koblitz curves. However, their proof of sublinearity has some flaws. In [13], the authors have proved that a sublinear algorithm is indeed possible using three bases, namely τ, $\tau - 1$ and $\tau^2 + \tau + 1$. Their software and hardware implementations using two bases τ and $\tau - 1$ are fast enough to give the feeling of a sublinear algorithm, but it lacks a theoretical proof. In [16], authors have used the precomputations to obtain further speed-ups. In this work, we represent the scalar using a generalization of DBNS representation, namely, multibase number representation. The exponent scalar is represented as a sum/difference of products of powers of 2, 3 and 5.

Our Contributions: The main contribution of this work is a set of two formulas for computing 5-fold ($5P$) of an elliptic curve point P, one for curves over binary fields and the other for curves over prime fields. These formulas can be used to compute the scalar multiples using quinary or DBNS expansion (using 2,5 or 3,5 as bases) of the scalar. We also generalize the algorithm used to compute scalar multiplication in double base [12] to accommodate a third base, namely, 5. Thus, the proposed scalar multiplication algorithms use a representation of the scalar as sum/difference of product of powers of 2, 3 and 5. Experimental evidences indicate it is faster than all scalar multiplication algorithms known so far for general curves over large prime fields. For general curves over binary fields, our algorithm performs quite competitively.

2 Background

In this section, we briefly outline the materials used as a prerequisite for this work. Interested readers can consult the cited works to check details.

2.1 ECC

In this section, we give a brief overview of elliptic curve cryptography. Details can be found in [3,4,5,20].

Definition. An elliptic curve E over a field K is defined by an equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in K$, and $\Delta \neq 0$, where Δ is the discriminant of E.

Applying admissible changes of variables, the Weierstrass equation (1) can be simplified. Over prime fields, $K = F_p$, of large characteristic ($\geq 2, 3$), the equation (1) can be simplified to

$$y^2 = x^3 + ax + b, \tag{2}$$

where $a, b \in K$ and $\Delta = 4a^3 + 27b^2 \neq 0$.

Over binary fields $K = F_{2^m}$, the non-supersingular curves are used for cryptography, whose Weierstrass equation can be simplified to the form:

$$y^2 + xy = x^3 + ax^2 + b, \tag{3}$$

where $a, b \in K$ and $\Delta = b \neq 0$.

The set $E(K)$ of rational points on an elliptic curve E defined over a field K forms an abelian group, under the operation (denoted additively) defined by the secant and tangent law. The special point $\mathcal{O}$, called the *point at infinity* plays the role of identity in this group.

The most natural representation for a point on an elliptic curve group is the affine representation, i.e., by an ordered pair of field elements satisfying the equation of the curve. However, group operations in affine representation require field inversions, which are the most expensive among field operations. To avoid inversions, several point representations in homogeneous (projective) coordinates have been proposed in the literature. The choice of a coordinate system for point representation in the elliptic curve group largely depends upon the so-called $[i]/[m]$-ratio, the ratio between the cost of a field inversion to that of a field multiplication. It is generally assumed that for binary fields $3 \leq [i]/[m] \leq 10$ and but it is significantly higher (30 or more) for prime fields [18]. Therefore, in this paper we consider affine ($\mathcal{A}$) coordinates for curves defined over binary fields and Jacobian ($\mathcal{J}$) coordinates, where the point $P = (X : Y : Z)$ corresponds to the point $(X/Z^2, Y/Z^3)$ on the elliptic curve for curves defined over prime fields.

To denote cost of field operations, we will use $[i]$, $[s]$ and $[m]$ to denote the cost of one inversion, one squaring and one multiplication respectively. We shall always neglect the cost of field additions. Also, over binary fields, we will neglect squarings as they are almost free (if normal bases are used) or of negligible cost (linear operation) (see [21] for more details). Moreover, over large prime fields, we will assume that $[s] = 0.8[m]$.

For curves over binary fields, we will use several elliptic curve group operations along with the quintupling operation presented in Section 3. These formulas have been listed in Table 1. We have included only those algorithms which will be used in this work. One operation needs a special mention: a repeated doubling formula (w-DBL) for these curves, originally proposed by Guajardo and Paar

Table 1. Costs of various Elliptic Curve group operations. The costs for curves over binary fields ($E(F_{2^m})$) are in affine coordinates. Those for curves over prime fields ($E(F_p)$) are in Jacobian coordinates.

Operation	Output	For $E(F_{2^m})$		For $E(F_p)$	
		proposed	Cost	proposed	Cost
DBL(P)	$2P$	–	$1[i] + 2[m]$	–	$6[s] + 4[m]$
ADD(P, Q)	$P + Q$	–	$1[i] + 2[m]$	–	$4[s] + 12[m]$
mADD(P, Q)	–	–	–	[9]	$3[s] + 8[m]$
w-DBL(P)	$2^w P$	[11]	$1[i] + (4w - 2)[m]$	[22]	$4w[m] + (4w + 2)[s]$
DA(P, Q)	$2P \pm Q$	[7]	$1[i] + 9[m]$	–	-
TPL(P)	$3P$	[7]	$1[i] + 7[m]$	[12]	$10[m] + 6[s]$
w-TPL	$3^w P$	–	–	[12]	$10w[m] + (6w - 5)[s]$
TA(P, Q)	$3P \pm Q$	[7]	$2[i] + 9[m]$	-	-

in [19] and subsequently improved by Lopez and Dahab in [11], which requires just one inversion to compute $2^w P, w \geq 1$.

For curves over prime fields of large characteristics, we will use Jacobian coordinates ($\mathcal{J}$). The following formulas for group arithmetic are available to us: DBL, w-DBL, TPL, w-TPL and ADD, which compute $2P, 2^w P, 3P, 3^w P$ and $P + Q$ respectively. Also, if the base point is given in affine coordinates ($Z = 1$), then the cost of the so-called *mixed addition* (mADD) ($\mathcal{J} + \mathcal{A} \rightarrow \mathcal{J}$) requires fewer computations than generic addition. Also, DBL and TPL are less expensive when $a = -3$ in (2). In Table 1, we summarize the complexity of these different elliptic curve formulas.

All ECDLP based cryptographic primitives, like encryption, decryption, signature generation and verification, need the operation of scalar multiplication. Given an integer d and an elliptic curve point P, it is the operation of computing dP. Efficiency of the scalar multiplication depends largely upon efficiency of the algorithms used for group arithmetic and representation of the scalar. In this work, we present two new algorithms for efficient group arithmetic and a new representation of the scalar using three bases. This combination considerably accelerates the computation of scalar multiplication in ECC.

2.2 Multibase Representation of an Integer

Let k be an integer and let $\mathcal{B} = \{b_1, \cdots, b_l\}$ be a set of "small" integers. A representation of k as a sum of powers of elements of $\mathcal{B}$ ($\Sigma_{j=1}^{m} s_j b_1^{e_{j1}} \cdots b_l^{e_{jl}}$, where s_j is sign) is called a multibase representation of n using the base $\mathcal{B}$. The integer m is the length of the representation. Double base representation or double base number system (DBNS) [14,15,12] is a special case with $|\mathcal{B}| = 2$. In the current article we are particularly interested in multibase representations with $\mathcal{B} = \{2, 3, 5\}$.

The double base number system is highly redundant. Also, these representations are very short in length. The multibase representations are even shorter and more redundant than the DBNS. The number of representations of n grows

Table 2. Number of multibase representation of small numbers using various bases

n	$\mathcal{B} = \{2,\ 3\}$	$\mathcal{B} = \{2,\ 5\}$	$\mathcal{B} = \{2,\ 3,\ 5\}$	$\mathcal{B} = \{2,\ 3,\ 5,\ 7\}$
10	5	3	8	10
20	12	5	32	48
50	72	18	489	1266
100	402	55	8425	43777
150	1296	119	63446	586862
200	3027	223	316557	4827147
300	11820	569	4016749	142196718

very fast in the number of base elements. For example, 100 has 402 DBNS representations (base 2 and 3), 8425 representations using the bases 2, 3 and 5 and has 43777 representations using the bases 2, 3, 5, and 7 (considering only positive summands, i.e. $s_j = 1$). The number of representations for some small integers n have been provided in Table 2. The multibase representation are very sparse also. One can represent a 160 bit integer using around 23 terms using $\mathcal{B} = \{2,\ 3\}$ and around 15 terms using $\mathcal{B} = \{2,\ 3,\ 5\}$ (see [14] for a result on length of DBNS representations).

In this article, unless otherwise stated, by a multibase representation of n we mean a representation of the form

$$n = \Sigma_i s_i 2^{b_i} 3^{t_i} 5^{q_i}$$

where $s_i = \pm 1$. We will refer to terms of the form $2^a 3^b 5^c$ as 3-integers. A general multibase representation, although very short, is not suitable for a scalar multiplication algorithm. So we are interested in a special representation with restricted exponents.

Definition: A multibase representation $n = \Sigma_i s_i 2^{b_i} 3^{t_i} 5^{q_i}$ using the bases $\mathcal{B} = \{2,\ 3,\ 5\}$ is called a *step* multibase representation (SMBR) if the exponents $\{b_i\}$, $\{t_i\}$ and $\{p_i\}$ form three separate monotonic decreasing sequences.

Needless to mention, an integer n has several SMBR's, the simplest one being the binary representation. If n is represented in SMBR, then we can write it using Horner's rule and an addition chain (like Double-base chain in [12]) for scalar multiplication can easily be developed.

Conversion to SMBR. An integer can be converted to a multibase representation using Greedy Algorithm:

Greedy Algorithm
`while(`k `>0)`
  `let z be the largest number` $2^b 3^t 5^p \leq k$`;`
  `output` (b, t, p)
  `replace` k `by` $k - z$
`endwhile`

The greedy algorithm produces near canonical (shortest) representations. We can implement the approximation step of the algorithm by using a three index array

$T[0..max2, 0..max3, 0..max5]$, where the array element $T[i, j, k]$ is $2^i 3^j 5^k$ and $max2$ $max3$ $max5$ are maximum possible powers of 2, 3, and 5 respectively. To represent a 160 bit integer, if one chooses $max2 = 160$, $max3 = \log_3 160 \approx 103$ and $max5 = \log_5 160 \approx 70$. greedy algorithm returned multibase representations with 15 terms on the average. Although, these representations are very sparse, they are not in SMBR. Algorithm mGreedy as described in below converts an integer into SMBR. mGreedy terminates because k gets reduced in each iteration.

Algorithm 1. mGreedy Algorithm for Conversion into SMBR

Input: k a positive integer; $max2, max3, max5 > 0$, the largest allowed binary, ternary and quinary exponents and the array $T[0..max2;\ 0..max3;\ 0..max5]$.

Output: The sequence $(s_i, b_i, t_i, p_i)_{i>0}$ such that $k = \sum_{i=1}^{m} s_i\, 2^{b_i}\, 3^{t_i}\, 5^{p_i}$, with $b_1 \geq \ldots \geq b_m \geq 0$, $t_1 \geq \ldots \geq t_m \geq 0$, $p_1 \geq \ldots \geq p_m \geq 0$.

1: $s \leftarrow 1$
2: **while** $k > 0$ **do**
3: for(b=0 to max2, t=0 to max3, p=0 to max5)
$z = T[b, t, p]$, the best approximation of k
4: **print** (s, b, t, p)
5: $max2 \leftarrow b, \quad max3 \leftarrow t, \quad max5 \leftarrow p$
6: **if** $k < z$ **then**
7: $s \leftarrow -s$
8: $k \leftarrow |k - z|$

Improving Performance of mGreedy. Algorithm mGreedy can be improved on two fronts: we can modify it to (1) obtain shorter representations and (2) run faster.

Obtaining Shorter Representations. Looking at the outputs of mGreedy one observes that the average length of the representations become higher because in some of the representations are lengthy. In these lengthy representations, one observes that, one or two of the three exponents in the leading term is (are) very small. If a particular exponent in the leading term is small, it becomes 0 very quickly and the representation reduces to a DBNS representation thereafter. If two of the exponents become 0 very quickly, then the representation even degenerates to a single base representation. We can overcome this shortcoming of mGreedy as described below.

Let c_1, c_2 and c_3 be three fractions less than 1. Let $x = 2^b 3^t 5^p$ be the best approximation for k in some iteration. Then in the next iteration mGreedy replaces $max2$ by b, $max3$ by t and $max5$ by p and searches for the best approximation for $k - x$ in $T[0..max2; 0..max3; 0..max5]$ again. Instead of searching the array $T[\,]$, from $[0, 0, 0]$ to $[max2, max3, max5]$ we now restrict the lower limit to $[c_1 \times max2, c_2 \times max3, c_3 \times max5]$. This does not allow any exponent to become very small at once and prevents the representation from degenerating into a single or double base format. Also, the new algorithm runs faster as the search space in each iteration is smaller than the unrestricted version. With this restriction, the mGreedy (with $max2 = 160,\ max3 = 103,\ max5 = 70, c_1 = .4. c_2 = .3$;

$c3 = .25$) returns an SMBR of average length less than 30 terms for integers of 160 bits (almost 20 % shorter).

Improving Run Time. Algorithms Greedy and mGreedy are search-based algorithms. They work by searching for the best approximation for the current value of k in the table $T[\,]$. The table $T[\,]$ contains $max2 \times max3 \times max5$ entries. In each iteration, Greedy chooses the best approximation for the current value of k in the table from $[0, 0, 0]$ to $[max2, max3, max5]$. The search space remains the same for each iteration. Algorithm mGreedy substitutes the values of $max2, max3, max5$ in each iteration by a smaller value and hence the search space becomes smaller each time. The measure described in the last paragraph to generate the shorter representations even further restricts the search space by raising the lower limit of the search space to $[c_1 \times max2, c_2 \times max3, c_3 \times max5]$ from $[0, 0, 0]$.

Although, the choice $max2 = 160, max3 = 103$ and $max5 = 70$ returns very short representation, the table $T[\,]$ becomes very large. Construction of the table and table look-up make the conversion slow. But, it is observed that smaller choices like $max2 = 84, max3 = 36, max5 = 16$ speed up the conversion process dramatically and the the length of the representation goes up by 1 or 2 terms. Therefore, in all practical purposes, smaller values can be used. If the table $T[\,]$ can be precomputed and stored, the conversion becomes almost instantaneous.

Most of the exponent integers used in scalar multiplication are randomly generated ones. One can generate a random integer directly in SMBR form to get rid of the conversion process altogether. Also, in the situation where the exponent is known beforehand, (like generating elliptic curve digital signature on a messgae), then the known exponent can be converted into the desired format offline and stored to be used when required. At least, in such applications, our scalar multiplication algorithm will perform much better than others.

We investigated on two more types of representations using three bases 2, 3 and 5: (1) SMBR with small anomalies and (2) SMBR with non-trivial digits.

SMBR with Small Anomalies: In this type of representation, the powers of 2, 3 and 5 form monotonic decreasing sequences except for some small deviations in some terms. Let w_1, w_2 and w_3 be the small permissible anomalies for the binary, ternary and quinary exponents respectively. Then a multibase representation $\Sigma_i s_i 2^{b_i} 3^{t_i} 5^{p_i}$ is a step representation with (w_1, w_2, w_3)-anomalies if $\{b_i\}$, $\{t_i\}$, and $\{p_i\}$ form monotonic decreasing sequences with a few exceptional terms for which $|b_i - b_{i-1}| \leq w_1$ or $|t_i - t_{i-1}| \leq w_2$, $|p_i - p_{i-1}| \leq w_3$ hold. Such representations can be used for scalar multiplication if the points $2^a 3^b 5^c$ for $0 \leq a \leq w_1, 0 \leq b \leq w_2, 0 \leq c \leq w_3$ can be precomputed and stored (see [16]). By choosing w_i's to be as small as 2, it was seen that the length of a MBNS representation can be made quite shorter (24-25 terms).

SMBR with Non-trivial Digits: So far we have considered representations $\Sigma_i s_i 2^{b_i} 3^{t_i} 5^{p_i}$, where $s_i \in \{1, 0, -1\}$. Let $\mathcal{D} = \{7, 11, 13, 17, 19, 23, 29, 31, \cdots\}$ be set of integers relatively prime to 2, 3, and 5. Let $\mathcal{D}_j$ be the set of the first j integers from $\mathcal{D}$. Let us consider the MBNS representation of the type $\Sigma_i s_i 2^{b_i} 3^{t_i} 5^{p_i}$

where $\pm s_i \in \mathcal{D}_j$. Such representations are also very short and can be used for scalar multiplication if the points sP for $s \in \mathcal{D}_j$ can be precomputed (see [16]).

3 Efficient Formulas to Compute $5P$

In this section we present two new quintupling formulas for elliptic curve points, one for curves over prime fields of large characteristic and the other for curves over fields of characteristic 2.

3.1 Point Quintupling in Curves over Binary Fields

As the $[i]/[m]$ ratio is known to be quite smaller in binary fields, affine elliptic curve group arithmetic is preferable. Hence we propose the new quintupling formula for such curves in affine coordinates. Also, it is routine work to translate them into other coordinates. Let $P(x, y)$ be a point on an elliptic curve given by Equation (3) over a binary field. Let the 5-fold of P be given by, $5P = (x_5, y_5)$. x_5 and y_5 can be computed as follows:

Let us define the following polynomials: $A = x^4 + x^3 + b$, $B = x^2(A + x^3)$, $C = A^3 + Bx^3$, $D = A^2(A^2 + B)$ Then,

$$x_5 = x + \frac{xBD}{C^2}$$
$$y_5 = y + x_5 + \frac{xAD^2}{C^3} + (x^2 + y)\frac{BD}{C^2}. \tag{4}$$

Given $P(x, y)$, let us check how much computation is required to compute $5P$ using the above formula. In Table 3 we list the subexpressions (and costs) required to compute x_5 and y_5. Let us consider the efficiency of the proposed formula. As this is the first point quintupling formula for curves over binary fields, we do not have any previous formula to compare performance. We can compute $5P$ as $2(2P) + P$. Using the generic ADD and DBL, it will cost 3 inversions. We can reduce one inversion by using composite formula double-and-add (DA) (see [7]). Using DA, computing $5P$ costs $2[i] + 11[m]$. If we compute $2P$ first and apply triple-and-add (TA)(see [7]) to P and $2P$, $(3 \times P + 2P)$, then the cost would

Table 3. Cost of the quintupling formula for curves over binary fields

Expression	Cost	Expression	Cost
A	$(2[s] + 1[m])$	B	$(1[m])$
C	$(1[s] + 2[m])$	D	$(1[m])$
$1/C$	$(1[i])$	$1/C^2$	$1[s]$
$\frac{BD}{C^2}$	$(2[m])$	x_5	$(1[m])$
$1/C^3$	$(1[m])$	$\frac{xAD^2}{C^3}$	$(3[m] + 1[s])$
y_5	$(1[m])$		
Total: $1[i] + 5[s] + 13[m]$			

again involve 3 inversions, as TA requires 2 inversions. Using the repeated doubling formula proposed in [11] and ADD, it costs $2[i] + 8[m]$. So, the proposed formula is more efficient than all these methods if $[i]/[m]$ ratio is 5 or more. In the next section, we provide the proof of correctness of the point quintupling explicit formula.

3.2 Proof of Quintupling in Curves over Binary Fields

For nonsupersingular curves over fields of characteristic 2, the division polynomials are given by

$$\begin{aligned}
\psi_1 &= 1 \\
\psi_2 &= x \\
\psi_3 &= x^4 + x^3 + a_6 \\
\psi_4 &= x^6 + a_6 x^2 \quad (= x^2(x^4 + a_6)).
\end{aligned} \tag{5}$$

The higher degree division polynomials can be obtained by applying the the following recurrence relations:

$$\begin{aligned}
\psi_{2n+1} &= \psi_{n+2}\psi_n^3 - \psi_{n-1}\psi_{n+1}^3 \\
\psi_2\psi_{2n} &= \psi_{n+2}\psi_n\psi_{n-1}^2 - \psi_{n-2}\psi_n\psi_{n+1}^2.
\end{aligned} \tag{6}$$

Using first of these recurrences with $n = 2$ and second one with $n = 3$, we get,

$$\begin{aligned}
\psi_5 &= \psi_4\psi_2^3 - \psi_1\psi_3^3 \\
&= \psi_4 x^3 - \psi_3^3 \\
\psi_6 &= (\psi_5\psi_3\psi_2^2 - \psi_1\psi_3\psi_4^2)/\psi_2 \\
&= (\psi_5\psi_3 x^2 - \psi_3\psi_4^2)/x.
\end{aligned} \tag{7}$$

Using the above division polynomials, we can derive the expressions for 5-fold of a point $P(x, y)$ on the curve using the following relation with $n = 5$:

$$[n]P = (x + \frac{\psi_{n+1}\psi_{n-1}}{\psi_n^2}, y + \psi_2 on + \frac{\psi_{n+1}^2\psi_{n-2}}{\psi_2\psi_n^3} + h_4\frac{\psi_{n+1}\psi_{n-1}}{\psi_2\psi_n^2})$$

where,

$$\psi_2 on = x + \frac{\psi_{n+1}\psi_{n-1}}{\psi_n^2}$$

and

$$h_4 = (x^2 + y).$$

If the point $5P$ has the coordinates (x_5, y_5), then it is an simple exercise to see that

$$\begin{aligned}
x_5 &= x + \frac{\psi_6\psi_4}{\psi_5^2} \\
y_5 &= y + x_5 + \frac{\psi_6\psi_6'\psi_3}{\psi_5^3} + (x^2 + y)(\frac{\psi_6'\psi_4}{\psi_5^2})
\end{aligned} \tag{8}$$

where, $\psi_6^{'} = \psi_6/x$. If we define polynomials as

$$\begin{aligned} A &= x^4 + x^3 + b \\ B &= x^2(A + x^3) \\ C &= A^3 + Bx^3 \\ D &= A^2(A^2 + B) \end{aligned} \tag{9}$$

then as can be checked, one has, $\psi_3 = A$, $\psi_4 = B$, $\psi_5 = C$ and $\psi_6 = xD$. Substituting these values in the Equations (8), we get the quintupling formula (4).

3.3 Point Quintupling in Curves over Large Prime Fields

In this section we present the point quintupling formula for elliptic curves over large prime fields. Proof of the formula for this case is very similar to the proof presented in Section 3.2. Hence we omit the proof in order to save space.

Let $P(X : Y : Z)$ be a point on the elliptic curve (2) over a prime field. Let $5P$ have coordinates $(X_5 : Y_5 : Z_5)$. Then X_5, Y_5 and Z_5 can be computed as follows:

$$\begin{aligned} X_5 &= XV^2 - 2YUW, \\ Y_5 &= Y(E^3(12VL^2 - V^2 - 16L^4) - 64TL^5), \\ Z_5 &= ZV, \end{aligned} \tag{10}$$

where, $T = 8Y^4$ $(2[s])$, $M = 3X^2 + aZ^4$ $(3[s] + 1[m])$, $E = 12XY^2 - M^2$ $(1[s] + 1[m])$, $L = ME - T$ $(1[m])$, $U = 4YL$ $(1[m])$, $V = 4TL - E^3$ $(1[s] + 2[m])$, $N = V - 4L^2$ $(1[s])$, $W = EN$ $(1[m])$.

The quantities in the parentheses are the cost of computing the corresponding subexpressions. Besides, computing X_5, Y_5 and Z_5 from these subexpressions require $1[s] + 3[m]$, $4[m] + 1[s]$ and $1[m]$. Hence, the cost of computing $5P$ by these formulas are $8[s] + 13[m] \approx 19.4[m]$ (if $Z = 1$) or $10[s] + 15[m] \approx 23[m]$ (if $Z \neq 1$).

This is the first explicit formula in literature to compute the multiplication-by-5 mapping for generic curves over arbitrary finite fields of characteristic > 3. Hence, we have no other formula to compare efficiency. Let us check the its efficiency vis-a-vis methods for computing $5P$. We can compute $5P$ by $2(2P)+P$ or by $3P+2P$. We can compute $5P$ by $2(2P)+P$ with $9[s]+17[m] \approx 24.2[m]$ (if P is in affine) or $14[s]+20[m] \approx 31.2[m]$ (if P is in Jacobian). Using the formula $2P + 3P$, we can compute $5P$ with $22[m] + 12[s] \approx 31.6[m]$ or $26[m] + 16[s] \approx 38.8[m]$ according as P is in affine or in Jacobian coordinates.

We will refer to the formula computing $5P$ as QPL. If $a = -3$, then $M = 3X^2+aZ^4$ can be computed as $3(X+Z^2)(X-Z^2)$ with a cost of $1[s]+1[m]$ saving $2[s]$. Hence like DBL and TPL, QPL is also cheaper over special curves with $a = -3$. Also, just as in case of (u-)DBL and (v-)TPL, an algorithm $w-$QPL to compute 5^wP can be designed which will be cheaper than w invocations of QPL. That is because for every invocation of QPL, one has to compute $Z_i = V_{i-1}Z_{i-1}$ and then compute $aZ_i^4 = aV_{i-1}^4Z_{i-1}^4$. This step should normally take $1[m]+2[s]$.

Table 4. Cost of the quintupling formulas for various types of elliptic curves

Curve	Condition	Cost
$y^2 = x^3 + ax + b$ over $K = F_p$	general	$10[s] + 15[m]$
	$a = -3$	$8[s] + 15[m]$
	after a QPL	$9[s] + 15[m]$
$y^2 = x^3 + ax^2 + b$ over $K = F_{2^m}$	general	$1[i] + 5[s] + 13[m]$

But as aZ_{i-1}^4 and V_{i-1}^2 are already computed in the last QPL operation, by saving these subexpressions, one can compute $aZ_i^4 = aV_{i-1}^4 Z_{i-1}^4$ by just one $[m]$ and one $[s]$, saving one $[s]$. We have summarized the cost of QPL in Table 4.

4 The Scalar Multiplication Algorithms

The scalar multiplication algorithms used in this work are generalizations to 3 bases of the algorithms used in [12]. Without going into routine details, we add that the computation can be immunized against side-channel attacks using standard techniques proposed in the literature. Algorithm 2 for curves over binary fields uses the group operations like ADD, DBL, w-DBL, DA (double-and-add), TA (triple-and-add) for efficient computation.

Algorithm 2. Scalar Multiplication for Curves over Fields of Even Characteristic

Input: An integer $k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i} 5^{p_i}$, with $s_i \in \{-1, 1\}$, and such that $b_1 \geq b_2 \geq \ldots \geq b_m \geq 0$, $t_1 \geq t_2 \geq \ldots \geq t_m \geq 0$ and $p_1 \geq p_2 \geq \ldots \geq p_m \geq 0$ and a point $P \in E(F_q)$

Output: the point $kP \in E(F_q)$

1: $Z \leftarrow s_1 P$
2: **for** $i = 1, \ldots, m-1$ **do**
3: $u \leftarrow b_i - b_{i+1}$
4: $v \leftarrow t_i - t_{i+1}$
5: $x \leftarrow p_i - p_{i+1}$
6: **if** $u = 0$ **then**
7: $Z \leftarrow (5^x Z)$
8: **if** $v \neq 0$ **then**
9: $Z \leftarrow 3(3^{v-1} Z) + s_{i+1} P$ //(TA used here)
10: **else**
11: $Z \leftarrow Z + s_{i+1} P$
12: **else**
13: $Z \leftarrow 5^x Z$
14: $Z \leftarrow 3^v Z$
15: $Z \leftarrow 2^{u-1} Z$
16: $Z \leftarrow 2Z + s_{i+1} P$ //(DA used here)
17: **Return** Z

In Algorithm 2 we describe the proposed scalar multiplication method to be used in conjunction with multibase representation for curves over binary fields. Note that Algorithm 2 requires b_1 doublings, t_1 triplings and p_1 quintuplings. The number of additions is precisely the number of terms in the expansion of k in which both the binary and ternary exponents are zero. Otherwise, the addition is always carried out by invoking a composite operation like double-and-add (DA) or triple-and-add (TA). Thus we need very few additions for the computations.

We do not present the algorithm for scalar multiplication for curves over prime fields here. It is a generalization of the algorithm presented in [12] to the case of 3 bases.

5 Scalar Multiplication Results

A theoretical analysis of double (or higher) base number system is still eluding the researchers. We are also unable to provide theoretical proofs of efficiency of our scalar multiplication algorithms. We will present their average performance seen in applying them to a huge number (10^3 to 10^6) of randomly generated scalars.

We randomly generated 1 million 160-bit integers and stored them in a file. All the experiments were conducted by retrieving integers from this file, so that the same integers were used for all the experiments. This minimizes the bias in estimates due the use of different sets of integers for different scenarios. We present the results of our experiments in this section.

We present the experimental results in the Tables 5, 6, 7 below. In these tables, (i) *max2, max3, max5:* stand for maximum powers for 2, 3 and 5 allowed

Table 5. Costs of elliptic curves Scalar Multiplication for 160-bit multipliers. The values of c_1, c_2, c_3 have been chosen as $0.4, 0.3$ and 0.25 respectively.

max2	max3	max5	alen	F_p-Cost	F_{2^m}-Cost
160	103	69	30.35	1646.89[m]	96.67[i]+ 693.7[m]
100	85	45	31.56	1669.09[m]	101.6[i]+731.5[m]
90	75	35	32.52	1678.73[m]	108.8[i]+704.2[m]
85	60	25	32.78	1681.04[m]	112.6[i]+691.1[m]
85	38	18	31.44	1645.43[m]	113.0[i]+677.2[m]

Table 6. Costs of elliptic curves Scalar Multiplication for 160-bit multipliers. The values of c_1, c_2, c_3 have been chosen as $0.4, 0.3$ and 0.25 respectively.

$max2$	$max3$	$max5$	w_1	w_2	w_3	#Points	alen	F_p-Cost	F_{2^m}-Cost
84	36	16	1	0	0	1	31.01	1649.5[m]	97.7[i] + 676.5[m]
84	36	16	0	0	1	1	29.4	1606.4[m]	90.7[i] + 681.3[m]
84	36	16	1	0	1	3	28.2	1590.2[m]	88.8[i] + 681.8[m]
84	36	16	1	1	0	3	28.5	1597.1[m]	87.5[i] + 681.3[m]
84	36	16	0	1	1	3	25.9	1566.4[m]	85.5[i] + 680.3[m]
84	36	16	1	1	1	7	24.4	1552.3[m]	83.8[i] + 680.4[m]

Table 7. Costs scalar multiplication for 160-bit multipliers represented in three bases with larger digits. The values of c_1, c_2, c_3 have been chosen as $0.3, 0.3$ and 0.25 respectively. Column # Points indicates the number of points to be precomputed and stored.

$max2$	$max3$	$max5$	#Points	alen	F_p-Cost	F_{2^m}-Cost
84	36	16	1	25.67	1569.23[m]	87.3[i] + 672.07[m]
84	36	16	2	24.5	1534.13[m]	83.18[i] + 666.26[m]
84	36	16	3	22.86	1514.77[m]	81.17[i] + 662.75[m]
84	36	16	4	21.76	1496.29[m]	79.25[i] + 661.34[m]
84	36	16	5	21.14	1486.37[m]	77.98[i] + 660.74[m]

to occur in SMBR expansions, (ii) *alen:* means the average length of the SMBR expansions found. (iii) *cost:* means average cost of scalar multiplication for the randomly generated integers.

It is observed that choosing smaller values for $max2$, $max3$, $max5$ does not affect the cost drastically. Hence we recommend smaller values like $(85, 40, 20)$ to be used instead of $(160, 103, 70)$.

5.1 Scalar Multiplication Without Precomputation

Let us first consider the cost of scalar multiplication using 3 bases without any precomputation. We conducted several experiments using various values of $max2$, $max3$ and $max5$ and also various values of c_1, c_2 and c_3. In Table 5, we have presented some of the results. Observe that for both kinds of curves, the best results were obtained when the highest possible powers of $max2$, $max3$ and $max5$, i.e. 160, 103 and 70, were chosen. However for these values the conversion from binary to MBNS is the slowest as the search space for the greedy algorithm is very big. Also, it was found that the maximum powers of 2, 3 and 5 observed in these expansions were much smaller. So, we choose smaller values for $max2$, $max3$ and $max5$ and observed that in these cases not only is the conversion very fast, but also the results are also quite competitive.

5.2 Scalar Multiplication with Precomputations

We conducted a huge number of experiments for scalar multiplication in 3-base expansion using precomputations. As mentioned earlier, we considered two kinds of precomputations: (1) SMBR with small anomalies and (2) MBNS with non-trivial digits. In former case, we choose w_1, w_2, w_3 to be 0 or 1, requiring storage of 1 to 7 precomputed points. The representations obtained in this case are very sparse (24.4 terms with a storage of 7 points). Some typical results have been presented in Table 6.

Also, we conducted experiments using SMBR representations with non-trivial larger coefficients. We allowed the SMBR to use various digit sets, starting from $\mathcal{D}_1 = \{7\}$ to $\mathcal{D}_8 = \{7, 11, 13, 17, 19, 23, 29, 31\}$. Use of $\mathcal{D}_i$ requires storage of i points. It was found that the representations are even sparser than SMBR with

anomalies. For example, with storage of 7 points, the multibase representation of a 160 bit integer could be 19.87 terms on average. Also, the computation scalar multiplication is quite cheaper than previous cases. Some typical results have been presented in Table 7.

5.3 Comparison

Let us compare the performance of the proposed scalar multiplication scheme to some of the schemes existing in the literature. Some of the most recent scalar multiplication algorithms for general curves have been proposed in [7,12,16,17,23].

In [23], the authors have proposed an efficient scalar multiplication algorithm based on Montgomery's ladder. Their scheme does not require any precomputation and is secure against side-channel attacks. In [7], several formulas for efficient arithmetic have been proposed and a novel representation of the scalar in powers of 2 and 3 has been proposed, which is used for scalar multiplication. We refer to this scheme as binary/ternary scheme. In [12], the authors have proposed two schemes based on double base number systems and have obtained very good results. [16] has extended that work by considering the use of DBNS with precomputations. The authors have computed efficiency of their scheme and compared with several schemes with scalars of 200, 300, ... bits. In [17] the authors have proposed a new point tripling formula based on decomposition to 2 isogenies. They have pointed out efficiency of scalar multiplication schemes. We will compare our schemes with the methods proposed in these works.

Although our scheme in the current form is not secure against side-channel attacks [26,27], side-channel resistance can be attained by some routine work. For example, attacks like simple power analysis attacks can be resisted by using some schemes like side-channel atomicity proposed in [6] which has almost no performance penalty. Also, attacks similar to differential power attacks can be resisted using curve randomization [24] or point randomization [10] countermeasures, which have a fixed cost (less than $50[m]$).

Let us first consider the scalar multiplication schemes for curves over prime fields without precomputation. Let the size of the elliptic curve group be of 2^{160}-order. For such scenario, the scheme proposed in [23] requires $2638[m]$ to compute the scalar multiplication. The best scheme proposed in [12] requires $1863[m]$ using double base number system. Of the several schemes proposed in [17] using the efficient tripling formula proposed in same work, the best scheme for this scenario is sextuple and add method. This method requires $1957[m]$ for the special curves used by them and almost the same amount of computation for arbitrary curves. Our best scheme ($max2 = 160$, $max3 = 103$, $max5 = 70$) (see Table 5), takes only $1646[m]$ on average. Even for smaller values of $max2$, $max3$, $max5$, our schemes are clear winners.

If the system admits precomputation and storage of a few points, then we can resort to the two methods using SMBR with small anomalies or SMBR with non-trivial digit sets. In [17], the best performance reported for this scenario is $1623[m]$ with 8 points of storage with $3 - NAF_3$ method. As can be checked from Tables 6 and 7 the methods proposed in this work invariably perform better

Table 8. Average number of field operations using the binary, NAF, ternary/binary and DB-chain approaches for $n = 160$ bits, and $[i]/[m] = 8$

Algorithm	$[i]/[m] = 8$			Algorithm	$[i]/[m] = 8$		
	$[i]$	$[m]$	$\approx [m]$		$[i]$	$[m]$	$\approx [m]$
binary	240	480	2400	ternary/binary	129	787	1819
NAF	213	426	2130	DB-chain	114	789	1701
3-NAF	200	400	2000	This work[1]	97	693	1469
4-NAF	192	384	1920	This work[2]	113	677	1581

even with less storage.The best method proposed in [16] for 200 bit scalars is the DBChain method with 8 non-trivial coefficients ($\mathcal{S}_8$-DBChain). To compare our scheme with their method, we experimented with 200 bit scalars. With $max2 = 105$, $max3 = 60$, $max5 = 35$ and $c_1 = 0.4$, $c_2 = 0.3$, $c_3 = 0.25$, length of SMBR was seen to be 23.58, while length of the DBNS representation was reported to be 25.9. Also, for 200 bit scalars, SMBR based scalar multiplication took $1909.12[m]$ computation on average in comparison to $2019[m]$ reported in [16].

For curves over binary fields, the proposed schemes perform even better. We summarize the comparisons in Table 8. In the table, binary and NAF refer to the traditional Binary and NAF based double-and-add algorithms. The DB-chain method refers to the one proposed in [12] and binary/ternary refers to a method proposed in [7]. The last column of the table is approximate cost obtained by the number of inversion to the $[i]/[m]$-ratio and adding the number of multiplication to it. The method proposed in this work outperforms every known scheme even without any precomputations or storage for precomputed points. The scheme can be further improved using precomputations (see Tables 5 and 6).

6 Conclusion

In this work we have presented two efficient formulas for point quintupling in ECC over binary and prime fields. Also, we have proposed two scalar multiplication algorithms to take advantage of the proposed formulas. These algorithms use a multibase representation of the scalar using 2, 3 and 5 as bases. Also, we have dealt with the situation where the system admits precomputation and storage of some precomputed points. Our empirical results indicate that all the proposed schemes, with or without precomputation, outperform the corresponding best scalar multiplication schemes previously known.

Acknowledgement. The first author is thankful to Fields Institute, Toronto, Canada for partially supporting this work. Part of this work was done when the first author was visiting Fields Institute in September 2006. Also, the authors are thankful to P. Rozenhart and E. Roettger for going through the article and providing some valuable inputs.

References

1. Avanzi, R.M., Sica, F.: Scalar Multiplication on Koblitz Curves using Double Bases. Tech Report. Available at http://eprint.iacr.org/2006/067
2. Avanzi, R.M., Dimitrov, V., Doche, C., Sica, F.: Extending Scalar Multiplication to Double Bases. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 130–144. Springer, Heidelberg (2006)
3. Cohen, H., Frey, G. (eds.): Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press, Boca Raton (2005)
4. Blake, I.F., Seroussi, G., Smart, N.P.: Elliptic Curves in Cryptography. Cambridge University Press, Cambridge (1999)
5. Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curves Cryptography. Cambridge University Press, Cambridge (2005)
6. Chevalier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. IEEE Transactions on Computers 53(6), 760–768 (2004)
7. Ciet, M., Lauter, K., Joye, M., Montgomery, P.L.: Trading inversions for multiplications in elliptic curve cryptography. Designs, Codes and Cryptography 39(2), 189–206 (2006)
8. Ciet, M., Sica, F.: An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 171–182. Springer, Heidelberg (2005)
9. Cohen, H., Miyaji, A., Ono, T.: Efficient Elliptic Curve Exponentiation Using Mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
10. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptography. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
11. Dahab, R., Lopez, J.: An Improvement of Guajardo-Paar Method for Multiplication on non-supersingular Elliptic Curves. In: Proceedings of the XVIII International Conference of the Chilean Computer Science Society (SCCC 1998), Antofagasta, Chile, November 12-14, 1998, pp. 91–95. IEEE Computer Society Press, Los Alamitos (1998)
12. Dimitrov, V., Imbert, L., Mishra, P.K.: Efficient and Secure Elliptic Curve Point Multiplication Using Double Base Chain. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–79. Springer, Heidelberg (2005)
13. Dimitrov, V., Järvinen, K.U., Jacobson, M.J., Chan, W.F., Huang, Z.: FPGA Implementation of Point Multiplication on Koblitz Curves Using Kleinian Integers. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 445–459. Springer, Heidelberg (2006)
14. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: An algorithm for modular exponentiation. Information Processing Letters 66(3), 155–159 (1998)
15. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Theory and applications of the double-base number system. IEEE Transactions on Computers 48(10), 1098–1106 (1999)
16. Doche, C., Imbert, L.: Extended Double-Base Number System with applications to Elliptic Curve Cryptography. Tech Report, Conference version to appear in Indocrypt (2006), Available at http://eprint.iacr.org/2006/330
17. Doche, C., Icart, T., Kohel, D.: Efficient Scalar Multiplication by Isogeny Decompositions. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 191–206. Springer, Heidelberg (2006)

18. Fong, K., Hankerson, D., Lòpez, J., Menezes, A.: Field inversion and point halving revisited. IEEE Transactions on Computers 53(8), 1047–1059 (2004)
19. Guajardo, J., Paar, C.: Efficient Algorithms for Elliptic Curve Cryptosystems over binary fields. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 342–356. Springer, Heidelberg (2000)
20. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
21. Hankerson, D., Lòpez Hernandez, J., Menezes, A.: Software implementation of elliptic curve cryptography over binary fields. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 1–24. Springer, Heidelberg (2000)
22. Itoh, K., Takenaka, M., Torii, N., Temma, S., Kurihara, Y.: Fast implementation of public-key cryptography on a DSP TMS320C6201. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 61–72. Springer, Heidelberg (1999)
23. Izu, T., Takagi, T.: Fast elliptic curve multiplications resistant against side channel attacks. IEICE Transactions Fundamentals E88-A(1), 161–171 (2005)
24. Joye, M., Tymen, C.: Protections against differential analysis for elliptic curve cryptography – an algebraic approach. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 377–390. Springer, Heidelberg (2001)
25. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48(177), 203–209 (1987)
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. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
28. Miller, V.S.: Uses of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–428. Springer, Heidelberg (1986)
29. Tijdeman, R.: On the maximal distance between integers composed of small primes. Compositio Mathematica 28, 159–162 (1974)

Enforcing Confidentiality in Relational Databases by Reducing Inference Control to Access Control

Joachim Biskup and Jan-Hendrik Lochner

Fachbereich Informatik, Universität Dortmund, D-44221 Dortmund, Germany
{biskup,lochner}@ls6.cs.uni-dortmund.de

Abstract. Security in relational database systems pursues two conflicting interests: confidentiality and availability. In order to effect a compromise between these interests, two techniques have evolved. On the one hand, controlled query evaluation always preserves confidentiality, but leads to undecidable inference problems in general. On the other hand, access control features simple access decisions, but possibly cannot avoid unwanted information flows. This paper introduces a form of access control that, in combination with restricting the query language, results in an efficient access control mechanism under preservation of confidentiality. Moreover, we justify the necessity of our restrictions and give an outlook on how to use our result as building block for a less restrictive but still secure system.

Keywords: Access control, confidentiality, database security, inference control, information flow, policy, potential secrets, relational databases.

1 Introduction

Information systems are primarily intended for providing structured data to users. In practice, relational databases are often used to serve this purpose since they are well understood and appropriate for modelling facts from the real world. However, besides the availability of data, also confidentiality has to be considered in order to prevent unwanted disclosure of information. This information does not need to be represented in the database explicitly. Rather, inferences can cause such information disclosure as well.

Classical techniques for enforcing database security normally do not cope with the inference problem. E. g., access control only prevents explicitly represented secrets from being disclosed. Inference control mechanisms like controlled query evaluation by Biskup and Bonatti [10] can solve this problem, but unfortunately these mechanisms are computationally inefficient or even infeasible in general.

As an illustration of the inference problem imagine a fictitious company and suppose there is an employee called Smith working in the sales department. Moreover, suppose that employees in the sales department get a salary of $ 2500. From this "basic" information we can derive that Smith gets a salary of $ 2500, although we do not know this fact "directly". Such inferences become crucial

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 407–422, 2007.

when the "basic" information is public but the inferred information shall be kept confidential.

In this paper we introduce the policy based classification of databases. In combination with restricting the query language, this technique preserves confidential information on the one hand and is computationally efficient on the other hand. The need of restricting the expressiveness of the query language is indeed a drawback of our mechanism, but relaxing these restrictions possibly leads to a lack of confidentiality. This is an indication for inference control being substantially different from access control on a semantic level.

Indeed, restricting the query language is not acceptable from the perspective of a database user. Therefore we shall use our basic results in order to develop a secure model allowing the full relational calculus (or at least a significant fragment) as query language. A thorough formal investigation of this issue is out of the scope of this paper but we will give an outlook on future work.

The remainder of this paper is structured as follows. Section 2 gives an overview over literature about database security in general and the problem of inference in particular; furthermore, we roughly bring our contribution in line with the related work. Section 3 recalls the basics of the relational data model and gives some auxiliary definitions in the context of secure query evaluation. In Sect. 4, we introduce the policy based classification of databases, propose two algorithms for enforcing it, and prove its security. Moreover, we give a critical review of our method, including the problem of expressiveness, and outline some perspectives for future research. In Sect. 5, we conclude our work.

2 Related Work

A general overview over the inference problem in different contexts can be found in the work of Farkas and Jajodia [19]. Fernández-Medina and Piattini [20] as well as Bertino and Sandhu [4] state and analyze requirements in the field of database security.

Access control is investigated by many authors. The main concepts and numerous extensions can be found in popular textbooks on computer security, e. g. [5,18,21]. Discretionary access control (DAC) as well as mandatory access control (MAC) operates on the actual data, and classification information is (at least conceptually) directly attached to this data, or its containers, respectively.

The general problem with discretionary access control is that the responsibility of correctly assigning access rights is with the owner of an object. In the case of relational databases the security administrator has to set appropriate access rights in order to prevent information disclosure by inference.

The mandatory approach employs system-wide policies on classified data according to a security model. General overviews on security models are given by McLean [24,25] and Sandhu [30]. The most known security model is the Bell-LaPadula model [3]. Roughly, in this model every user is assigned a clearance from a lattice of security levels and will only be allowed to read an object if the classification of this object is dominated by the clearance of the user. Moreover,

the user will only be allowed to write an object if his clearance is dominated by the classification of the object. This approach, although critically discussed e. g. by McLean et al. [23] and by Nicomette and Deswarte [26], is able to prevent unwanted information flows caused by sequences of read and write operations.

Early work on mandatory access control in the context of relational databases was done by Lunt et al. [22] by proposing a formal security model. Among many other topics, subsequent work studies how to consistently declare the classifications of structured objects that are bound to constraints, in order to prevent unwanted inferences, see e. g. Olivier and von Solms [27], Cuppens and Gabillon [14,15], and Dawson et al. [16,17]. Basically, the classifications of composed or derivable information should be dominated by the classifications of all the constituents or presuppositions, respectively.

Examples of further work on confidentiality in databases include the technique of query rewriting, proposed by Stonebreaker and Wong [32] and recently reconsidered by Rizvi et al. [28], and the Disclosure Monitor of Brodsky et al. [13].

Controlled query evaluation (CQE) for logical databases has been developed by Biskup and Bonatti [6,7,8,9,10], based on the work of Sicherman et al. [31] and Bonatti et al. [12], respectively. Biskup and Bonatti [11] propose CQE variations for open queries in relational databases. CQE guarantees perfect confidentiality but is, unfortunately, computationally infeasible in general.

In contrast to most previous work we focus on the information to be protected according to the security policy rather than on the actual data in the database instance. Furthermore we state sufficient conditions so that the security administrator has not to care about harmful inferences, since they cannot occur in our setting. Since we propose a static mechanism for enforcing security policies we are also able to state efficient algorithms.

Further information about relational query languages and their computational complexity can be obtained e. g. from [1].

3 Preliminaries

We begin with some definitions concerning relational databases. Afterwards, a framework for preserving confidentiality in relational databases is introduced.

3.1 Relational Databases

Definition 1 (Relation schemas and instances). *A* relation schema $RS = \langle R, \mathcal{U}, \Sigma \rangle$ *consists of a* relation symbol R*, a finite set of* attributes $\mathcal{U} = \{A_1, \ldots, A_n\}$*, and a finite set of semantic constraints* Σ*. Const denotes an infinite set of* constants*. An* instance r *of a relation schema is a finite Herbrand interpretation*[1] *of the schema, considering the relation symbol as a predicate. A* tuple *is denoted by* $\mu = R(a_1, \ldots, a_n)$ *where* $a_i \in Const$*. If a formula* ϕ *of an appropriate language is true in* r*, we write* $r \models_{\mathrm{M}} \phi$*.*

[1] An Herbrand interpretation I interprets constants by themselves, $I(c) = c$ for every $c \in Const$. I is called finite if every predicate symbol is interpreted by a finite set.

Remark. On the database level we identify an instance r with the set of tuples μ such that r makes μ true. If $r \models_{\mathrm{M}} \mu$ we shortly say "μ is in r".

Definition 2 (Functional dependencies). *Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{U}$ be sets of attributes. An instance r satisfies the* functional dependency (fd) *$\mathcal{A} \to \mathcal{B}$ iff any two tuples in r agreeing on the values of the attributes of $\mathcal{A}$ also agree on the values of the attributes of $\mathcal{B}$. $\mathcal{A} \to \mathcal{B}$ is called* trivial *if $\mathcal{B} \subseteq \mathcal{A}$.*

In the following, we assume a database consisting of exactly one relation schema $RS = \langle R, \mathcal{U}, \Sigma \rangle$ with Σ containing only functional dependencies. The schema of the database is then determined by RS and *Const*. Furthermore, let the domain of every $A_i \in \mathcal{U}$ be *Const*.

Logic related terms and concepts are to be seen under the restrictions of Definition 1. As a result, arguing about the set of all interpretations (as in the usual definition of the $\models$-operator) in fact means arguing about the set of all finite Herbrand interpretations. Moreover, different constants are always interpreted differently by the Herbrand interpretation, becoming important when arguing about inconsistency. Finally we apply the *closed world assumption*, saying that for any database instance r and any tuple μ, if μ is not explicitly represented in r we assume that μ is false in r. We will use this notion in a wider sense, considering not only tuples but arbitrary suitable formulas.[2]

Note that the "single relation restriction" only serves as a simplification of our investigations and could easily be abolished, whereas the "functional dependencies only restriction" is substantial. Since we do not consider multilevel security, there is the same confidentiality policy for every user. Therefore, we are able to model the users of our system as a single user.

In order to retrieve information from the database, a user expresses queries in an appropriate language. In Definition 3, we define a simple fragment of the relational calculus, enabling the user to express closed variable-free queries without sentential connectives. The common procedure of evaluating closed queries w. r. t. a database instance is recalled in Definition 4.

Definition 3 (Query language). *The* query language $\mathscr{L}_{\mathrm{q}}$ *is defined by*

$$\mathscr{L}_{\mathrm{q}} := \{R(a_1, \ldots, a_n) \mid a_i \in \mathit{Const}\}\,.$$

Definition 4 (Ordinary query evaluation). *The* ordinary query evaluation *eval maps a query $\Phi \in \mathscr{L}_{\mathrm{q}}$ on a truth value, dependent on the database instance r:*

$$\mathit{eval}(\Phi)(r) := r \models_{\mathrm{M}} \Phi.$$

An alternative version eval^* maps Φ on Φ or $\neg\Phi$, respectively:

$$\mathit{eval}^*(\Phi)(r) := \texttt{if}\ r \models_{\mathrm{M}} \Phi\ \texttt{then}\ \Phi\ \texttt{else}\ \neg\Phi.$$

[2] Biskup and Bonatti [11] call the interpretations characterized by the properties of Definition 1 *DB-interpretations* and consider a particular *DB-implication* $\models_{DB}$.

3.2 Secure Evaluation of Queries

A *confidentiality policy* describes which information a user should not know. Potential secrets, as defined below, form a special kind of confidentiality policies, consisting of formulas a user should not know in case they are true in the actual database instance. We use a fragment of the positive existential calculus without free variables to define potential secrets:

Definition 5 (Potential secrets). *Let Var be an infinite set of variables. The language of* potential secrets $\mathscr{L}_{\text{ps}}$ *is defined by*

$$\begin{aligned}\mathscr{L}_{\text{ps}} := \{\, &(\exists X_1)(\exists X_2)\ldots(\exists X_l)R(v_1,\ldots,v_n) \mid 0 \le l \le n,\\ &X_i \in \mathit{Var}, v_i \in \mathit{Var} \cup \mathit{Const}, \{X_1,\ldots,X_l\} \subseteq \{v_1,\ldots,v_n\},\\ &v_i \in \mathit{Var} \implies v_i = X_j \text{ for some } j \in \{1,\ldots,l\},\\ &v_i, v_j \in \mathit{Var} \implies v_i \neq v_j \,\}.\end{aligned}$$

Remark. Note that $\mathscr{L}_{\text{q}}$ is a subset of $\mathscr{L}_{\text{ps}}$. Therefore, the language of all queries and all potential secrets can be denoted by $\mathscr{L}_{\text{ps}}$ for short. Sets of potential secrets are usually denoted by *pot_sec*.

Example 1. Consider the schema $RS_1 = \langle R_1, \{A, B, C\}, \{A \to BC\}\rangle$ and the instance $r_1 = \{\mu_1, \mu_2, \mu_3\}$ with $\mu_1 = R_1(a_1, b_1, c_1)$, $\mu_2 = R_1(a_2, b_1, c_2)$, and $\mu_3 = R_1(a_3, b_1, c_1)$. Moreover, let $\mathit{pot\_sec}_1 = \{\Psi_1, \Psi_2\}$ with $\Psi_1 = (\exists X_1)R_1(X_1, b_1, c_1)$ and $\Psi_2 = R_1(a_2, b_2, c_2)$. Ψ_1 shall prevent the user from discovering the combination $(B = b_1, C = c_1)$. An appropriate enforcement of this policy must keep μ_1, μ_3 secret, and even the critical parts of these tuples, as well as any further information disclosing the occurrence of $(B = b_1, C = c_1)$ in r_1. Ψ_2 is false in r_1 and therefore does not need not to be protected.

Although being designed for the control of a single user originally, we can extend the notion of potential secrets to lattice-based mandatory security models [30] as well. By labeling every potential secret Ψ_i with a classification c_i, we get a policy $\mathit{pot\_sec}^* = \{(\Psi_1, c_1), (\Psi_2, c_2), \ldots, (\Psi_m, c_m)\}$. A user u is assigned a clearance cl_u, meaning that u is only allowed to read information classified at most as high as cl_u. In return, u must not read information classified higher than or incomparable with cl_u. As a result, for u we obtain a "personal confidentiality policy" $\mathit{pot\_sec}_u = \{\Psi_i \mid (\Psi_i, c_i) \in \mathit{pot\_sec} \text{ and } c_i \not\leq cl_u\}$.

Biskup and Bonatti developed the technique of *controlled query evaluation (CQE)*, modifying the ordinary query evaluation by distorting answers if necessary to preserve confidentiality w. r. t. a given confidentiality policy (see [10] for an overview). We adopt their notion of security and apply it to the *modified query evaluation*, a generalized version of CQE.

Definition 6 (Modified query evaluation). *A* modified query evaluation *m_eval maps a query sequence $Q = \langle \Phi_1, \Phi_2, \ldots\rangle$, a database instance r and a confidentiality policy pot_sec on an answer sequence:*

$$\mathit{m\_eval}(Q)(r, \mathit{pot\_sec}) = \langle \mathit{ans}_1, \mathit{ans}_2, \ldots\rangle\,.$$

Definition 7 (Secure query evaluation). *Consider a (possibly infinite) query sequence* $Q = \langle \Phi_1, \Phi_2, \ldots \rangle$ *with* $\Phi_i \in \mathscr{L}_{\mathrm{q}}$ *and a confidentiality policy* $pot\_sec = \{\Psi_1, \ldots, \Psi_m\}$ *with* $\Psi_i \in \mathscr{L}_{\mathrm{ps}}$. *The modified query evaluation* m_eval *is* secure w. r. t. pot_sec *if for every finite prefix* Q' *of* Q, *for every* $\Psi \in pot\_sec$, *and for every instance* r_1 *of* RS *there exists an instance* r_2 *of* RS *satisfying the following properties:*

1. $m\_eval(Q')(r_1, pot\_sec) = m\_eval(Q')(r_2, pot\_sec)$;
2. $eval^*(\Psi)(r_2) = \neg\Psi$;

m_eval *is called* secure *if it is secure w. r. t. every* pot_sec.

Remark. Note that the first property in this definition demands the same answers of r_1 and r_2 under a fixed pot_sec. Consequently, this notion of security even allows the user to know the confidentiality policy.

4 Policy Based Classification of Databases

In this section we introduce our method of appropriately classifying relational databases to enforce efficient access control and guaranteeing the security property in the sense of Definition 7 at the same time. To appropriately represent a confidentiality policy in the context of relational databases we develop the concept of *classification instances*. For simplicity, we do not assume several classification levels but regard information as being secret or not.

4.1 Classified Databases

Formally, the classification of a database is an instance of a classification schema which is obtained from the schema of the database to be classified.

Definition 8 (Classification schemas). *The* classification schema *of the relation schema* $RS = \langle R, \mathcal{U}, \Sigma \rangle$ *is defined by* $RS^{\mathrm{C}} = \langle S, \mathcal{U}, \emptyset \rangle$ *and* $Const^{\mathrm{C}} = Const \cup \{\#\}$ *with* $\# \notin Const$, *where* S *denotes a new relation symbol.*

Definition 9 (Classification instances). *Let* pot_sec *be a set of potential secrets. The* classification instance s *w. r. t.* pot_sec *is defined as follows: For every* $\Psi = (\exists X_1) \ldots (\exists X_l) R(v_1, \ldots, v_n)$ *with* $\Psi \in pot\_sec$, *a (generalized) tuple* $S(v_1^*, \ldots, v_n^*)$ *is inserted into* s *where* $v_i^* = v_i$ *if* $v_i \in Const$ *and* $v_i^* = \#$ *otherwise. There are no further formulas in* s.

Example 2. Recall the set pot_sec from Example 1. The corresponding classification instance is $s = \{\mu_1^*, \mu_2^*\}$ with $\mu_1^* = S(\#, b_1, c_1)$ and $\mu_2^* = S(a_2, b_2, c_2)$.

With the classification instance a query can be determined as allowed or not allowed. Intuitively, a tuple in the classification instance represents a value combination to be kept secret. The symbol # denotes a placeholder, similar to a null value: the value is existing but irrelevant from the perspective of confidentiality.

If the constants in a query match with the constants of a tuple in the classification instance, this query asks for a secret and must therefore not be allowed. Note that even a partial matching may be sufficient to disallow a query: if all constants of the tuple (except #) occur in the query the answer must be refused, regardless of the other constants in the query. Thus, to formally define allowed queries, we have to introduce the notion of relevance at first.

Definition 10 (Relevance). *Let* $\chi_1, \chi_2 \in \mathscr{L}_{\text{ps}}$. $\chi_i(A)$ *denotes the value of attribute* A *in* χ_i. χ_1 *is called* relevant *for* χ_2 *iff for every* $A \in \mathcal{U}$ *it holds that* $\chi_1(A) \in \mathit{Const} \implies \chi_1(A) = \chi_2(A)$.

Definition 11 (Allowed queries). *A query* $\Phi = R(v_1, \ldots, v_n)$ *against an instance* r *is* allowed *w. r. t. a set of potential secrets pot_sec if in the classification instance* s *(w. r. t. pot_sec) there does not exist a tuple* μ^* *being relevant for* Φ.

The access control function defined below is a modified query evaluation in the sense of Definition 6.

Definition 12 (Access control). *Let* r *be an instance of* RS, s *the classification instance w. r. t. pot_sec, and* $Q = \langle \Phi_1, \Phi_2, \ldots \rangle$ *a (possibly infinite) query sequence with* $\Phi_i \in \mathscr{L}_{\text{q}}$. *The* access control function *ac is defined by*

$$ac(Q)(r, pot\_sec) := \langle ans_1, ans_2, \ldots \rangle$$

with

$$ans_i := \begin{cases} eval^*(\Phi_i)(r) & \text{if } \Phi_i \text{ is allowed w. r. t. } pot\_sec, \\ \mathit{mum} & \text{otherwise.} \end{cases}$$

Note that the access control decision is instance independent. Consequently, analogous to CQE with refusal [7], *ac* has to protect *every* $\Psi \in pot\_sec$, regardless of Ψ being true in r or not.

Example 3. Again, recall Example 1 and the classification instance s from Example 2. We examine the query sequence $Q_1 = \langle \Phi_{1,1}, \Phi_{1,2}, \Phi_{1,3} \rangle$ with $\Phi_{1,1} = R_1(a_4, b_1, c_1)$, $\Phi_{1,2} = R_1(a_2, b_1, c_2)$, and $\Phi_{1,3} = R_1(a_2, b_2, c_2)$. It is easy to see that $\Phi_{1,1}$ and $\Phi_{1,3}$ are not allowed according to Definition 11: Although $\Phi_{1,1}$ is false in r_1, it contains a critical part, namely the combination $(B = b_1, C = c_1)$; consequently, with μ_1^* there exists a tuple in s violating the condition of Definition 11. $\Phi_{1,3}$ is true in r_1 and to be kept secret because of μ_2^*. Finally, $\Phi_{1,2}$ is true in r_1 and furthermore allowed w. r. t. pot_sec_1. Thus, the answer sequence to Q_1 is $\langle \mathtt{mum}, R_1(a_2, b_1, c_2), \mathtt{mum} \rangle$.

4.2 Efficient Enforcement

Unlike inference control in general, for a single query our access control mechanism can be enforced efficiently. In the following we constructively show this by

sketching two algorithms. Considering the number n of attributes of the underlying database as a constant, the first algorithm has linear runtime w. r. t. the size m of the confidentiality policy, whereas the second algorithm enforces our mechanism even in logarithmic time. Since the ordinary query evaluation has to be carried out in either case, we do not consider its runtime but concentrate on the runtime of the access control mechanism.

Consider a query $\Phi \in \mathscr{L}_{\mathrm{q}}$, a database instance r of RS, a confidentiality policy pot_sec, and a corresponding classification instance s. Moreover, let n denote the number of attributes in RS (considered constant), $m = |pot\_sec|$, and $|r|$ the number of tuples made true by r.

Algorithm A (Linear Time): For every μ^* in s check, whether every attribute not instantiated by # in μ^* has the same value in the query Φ. If so, refuse the answer; otherwise, return $eval^*(\Phi)(r)$.

Since s contains as many formulas as pot_sec, the access control in Algorithm A has a linear runtime of $O(n \cdot m)$.

Algorithm B (Logarithmic Time): Suppose, the tuples in s are indexed by a B-tree T (see [2]). Construct the set P_Φ of formulas being relevant for Φ by successively replacing every subset of values in Φ with the special value #, e. g., if $\Phi = R(a, b)$, then $P_\Phi = \{R(a, b), R(\#, b), R(a, \#), R(\#, \#)\}$. For every $\rho \in P_\Phi$, check if ρ is in s by searching T. If any ρ occurs in T, refuse the answer; otherwise, return $eval^*(\Phi)(r)$.

Note that $|P_\Phi| = 2^n$. Furthermore, the height h of T (and thereby the worst-case search time) is bounded by $h \leq 2 + log_{k+1}\left(\frac{m}{2k}\right)$ where the number of entries in each node of T lies between k and $2k$ (cf. [2]). Since k is an implementation-dependent constant, h can be estimated by $O(log(m))$. Consequently, the access control in Algorithm B has a logarithmic runtime of $O(2^n \cdot log(m))$.

Provided that the tuples in r are also appropriately indexed by a B-tree, the ordinary query evaluation takes time $O(log(|r|))$. We sum up our results in the following proposition.

Proposition 1 (Efficient computability of ac). *There exists an algorithm for computing the access control function $ac(\langle\Phi\rangle)(r, pot\_sec)$ in time $O(\min\{n \cdot m, 2^n \cdot log(m)\} + log(|r|))$.*

4.3 Proof of Security

We now prove the policy based classification of databases secure in the sense of Definition 7. To this end, we have to do a little preliminary work in form of two lemmas.

Lemma 1. *Let χ and $\chi_1, \ldots, \chi_p$ be in $\mathscr{L}_{\mathrm{ps}}$. Then, the following conditions are equivalent:*

1. $\{\chi_1, \ldots, \chi_p\} \models \chi$.
2. *There exists a $\chi_i \in \{\chi_1, \ldots, \chi_p\}$ such that χ is relevant for χ_i.*

Sketch of Proof. The "only if"-case can be shown by contraposition constructing a witness instance r against condition 1, being a model for $\{\chi_1, \ldots, \chi_p\}$ but not for χ. For the "if"-case we simply employ the definition of relevance.

Lemma 2. *Let $\mathcal{S}$ be a finite, consistent set of formulas and $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ a partition of $\mathcal{S}$ satisfying the following properties:*

1. $\phi \in \mathcal{S}_1 \implies \phi \in \mathscr{L}_{\mathrm{q}}$.
2. $\phi \in \mathcal{S}_2 \implies \phi = \neg\phi'$ *with* $\phi' \in \mathscr{L}_{\mathrm{q}}$.
3. $\phi \in \mathcal{S}_3 \implies \phi$ *is an fd.*

Then, considering some $\Psi \in \mathscr{L}_{\mathrm{ps}}$, the following conditions are equivalent:

1. $\mathcal{S} \models \Psi$.
2. *There exists a* $\chi \in \mathscr{L}_{\mathrm{q}}$ *with* $\chi \in \mathcal{S}$ *and* $\chi \models \Psi$.

Sketch of Proof. We show the "only if"-case by contraposition using a witness instance r against condition 1. This instance contains exactly the formulas from $\mathcal{S}_1$ and can also be proved to be a model for $\mathcal{S}_2$ and $\mathcal{S}_3$, leading to $r \models_{\mathrm{M}} \mathcal{S}$. By the assumption of contraposition, $\mathcal{S}_1 \not\models \Psi$. Using Lemma 1, we then prove Ψ not being relevant for any $\phi \in \mathcal{S}_1$, yielding $r \not\models_{\mathrm{M}} \Psi$. The "if"-case is obviously satisfied.

Now, we are able to prove the security of our access control method.

Theorem 1 (Policy based classification of databases is secure). *Consider the instance r of RS, the confidentiality policy $pot\_sec \subset \mathscr{L}_{\mathrm{ps}}$, the (possibly infinite) query sequence $Q = \langle \Phi_1, \Phi_2, \ldots \rangle$ with $\Phi_i \in \mathscr{L}_{\mathrm{q}}$, and the classification instance s of r w. r. t. pot_sec. Let furthermore ac be the access control function according to Definition 12. Then, ac is secure w. r. t. pot_sec in the sense of Definition 7.*

Sketch of Proof. We consider a finite prefix $Q' = \langle \Phi_1, \ldots, \Phi_n \rangle$ of Q, a $\Psi \in pot\_sec$, and a set log, formed by Σ and the answers to the allowed queries in Q'. Lemma 2 is used to show that $log \not\models \Psi$. Hence, there exists a witness instance r' with $r' \models_{\mathrm{M}} log$ and $r' \not\models_{\mathrm{M}} \Psi$, thereby satisfying the properties of Definition 7.

The theorem shows the security of our access control w. r. t. a fixed Q, a fixed pot_sec, and a fixed r. However, the proof applies for arbitrary query sequences, confidentiality policies and database instances, resulting in the (general) security of the access control.

Corollary 1. *The access control function according to Definition 12 is secure in the sense of Definition 7.*

As a result, the restrictions demanded by the policy based classification make sure that potential inferences (due to functional dependencies) cannot actually be employed by the user. Regarding the user knowledge is no longer necessary for a secure query evaluation, enabling us to design a static (and therefore efficient) confidentiality enforcing mechanism. Clearly, there is a trade-off between efficiency and expressive power of the query language. We address this problem in the forthcoming subsection.

4.4 A Critical Review

We developed an access control method, preserving confidentiality in relational databases and being efficiently computable at the same time. Moreover, the security of our approach has been shown formally in Theorem 1—as far as we know, such a theorem has never been established for DAC/MAC. Indeed, we had to accept several restrictions to make our method work properly. The major drawback lies in the query language $\mathscr{L}_{\mathrm{q}}$. Compared to the full relational calculus, it is obviously less expressive.

However, from the perspective of a database user a substantial restriction of the query language is not reasonable. Thus, we see our contribution as a building block for a more expressive model and plan to enhance the query language to the full relational calculus in future work.

In doing so we will adapt the access control mechanism appropriately. Roughly speaking, we will consider complex queries as views over the base relations of the database. Following the proposal of Rosenthal et al. [29], we will allow access to a view if access to all "basic elements" of the view is allowed. More precisely, if a user is allowed to access the building blocks of a complex query, he should also be allowed to access the view defined by this query, since he could construct this view by appropriately composing the building blocks just as well. Moreover, if at least one of the building blocks must not be accessed by the user, also access to the view should be rejected as justified by the following example.

Example 4. Consider two simple queries $\Phi_1, \Phi_2 \in \mathscr{L}_{\mathrm{q}}$ and a complex query $\Phi_3 = \Phi_1 \vee \Phi_2$ being false in the database instance. Also suppose that the user must not learn the truth value of Φ_1 in the instance. If Φ_3 was allowed the user could infer the confidential truth value of Φ_1 by learning that Φ_3 is false.

Considering only closed variable-free queries we conjecture that this approach works quite straightforward. In the case of (free or bound) variables, things become a bit more complicated. We illustrate these difficulties with two additional examples.

Example 5. Suppose an alternative query language $\mathscr{L}_{\mathrm{q}}^{\exists} := \mathscr{L}_{\mathrm{ps}}$. We examine the schema $RS_2 = \langle R_2, \{A, B, C\}, \{A \to BC\}\rangle$ with the instance $r_2 = \{R_2(a_1, b_1, c_1)\}$ and the policy $pot\_sec_2 = \{(\exists X)R_2(X, b_1, c_1)\}$. Then, we have the classification instance $s = \{S(\#, b_1, c_1)\}$.

Now assume the query sequence $Q_2 = \langle \Phi_{2,1}, \Phi_{2,2}\rangle$ defined by $\Phi_{2,1} = (\exists X)$ $R_2(a_1, b_1, X)$ and $\Phi_{2,2} = (\exists X)R_2(a_1, X, c_1)$. Both $\Phi_{2,1}$ and $\Phi_{2,2}$ are allowed in the sense of Definition 11, leading to the answer sequence $\langle (\exists X)R_2(a_1, b_1, X), (\exists X)$ $R_2(a_1, X, c_1)\rangle$. However, utilizing the fd $A \to BC$, this information enables the user to infer $R_2(a_1, b_1, c_1)$ being true in r_2 and thus to disclose the potential secret.

As seen in this example, the advantage of our restrictive query language, namely that functional dependencies cannot be exploited for harmful inferences, gets lost if we allow the usage of existentially quantified variables. But also in the absence

EMP	pnr	name	dept	salary
	001	Jones	accounting	2000
	002	Jackson	sales	2500
	003	Stone	purchasing	2200
	004	Smith	sales	950
	005	Evans	advertising	3200

Fig. 1. Employees table

of functional dependencies confidentiality is possibly violated if admitting the full relational calculus as query language.

Example 6. Assume a table containing the personnel number, name, department, and salary of employees of a company (Fig. 1). Formally, this is an instance of the schema $RS_3 = \langle EMP, \{\text{pnr}, \text{name}, \text{dept}, \text{salary}\}, \Sigma \rangle$ with Σ being a set of functional dependencies. Furthermore, we want to hide the fact that Smith works in the sales department, leading to the confidentiality policy

$$pot\_sec_3 = \{(\exists X_1)(\exists X_2) EMP(X_1, \text{Smith}, \text{sales}, X_2)\} \quad .$$

For the sake of brevity we omit the respective classification instance. A user now poses the query sequence $Q_3 = \langle \Phi_{3,1}, \Phi_{3,2} \rangle$ with

$$\begin{aligned} \Phi_{3,1} = &(\forall X_1)(\exists X_2)(\exists X_3)(\exists X_4)(\exists X_5) \\ &(EMP(X_2, \text{Jackson}, X_1, X_3) \implies EMP(X_4, \text{Smith}, X_1, X_5)), \end{aligned}$$

and

$$\Phi_{3,2} = (\exists X_1)(\exists X_2) EMP(X_1, \text{Jackson}, \text{sales}, X_2) \quad .$$

At the first glance, neither $\Phi_{3,1}$ nor $\Phi_{3,2}$ has to be refused, since there is no obvious violation of the security policy: the critical combination (name: Smith, dept: sales) occurs in none of the building blocks of the queries. Nevertheless, if the systems answers both queries correctly (they are both true in the instance) the user can infer the secret. That is because the first answer tells him that in every department with an employee Jackson there also works an employee called Smith; the second answer then discloses the fact that there is an employee Jackson in the sales department. Consequently, there must also be an employee called Smith in the sales department.

When we admit variables in the query language—whether they are bound or even free—there is no obvious way to reduce access control to the basic parts of a query, since these parts are—if they actually contain variables—no longer elements from $\mathscr{L}_{\text{q}}$. A suitable substitution of the variables by constants is necessary to transfer our results to this kind of queries. A detailed investigation will be subject of future work.

5 Conclusion

In this paper, we proposed a method for enforcing confidentiality policies in relational databases. Moreover, we proved our method efficiently computable and secure under a restricted query language. We pointed out the necessity of a restriction by showing that more expressive languages might lead to the disclosure of secrets. Since a substantial restriction of the query language is not acceptable from the perspective of a database user, we finally roughly sketched an approach for more complex queries and pointed at the problems occuring with queries containing variables.

Our results indicate that the goals of (dynamic) inference control can be enforced by (static) access control only if we are willing to accept some kind of restriction. Future research should try to determine the minimal restriction being necessary to ensure confidentiality on the one hand and maximize availability and expressiveness of the query language on the other hand.

A minor drawback of our method is the fact that a security administrator has no means to comfortably declare a set of similar secrets. Extending the confidentiality policy language by free variables could solve this problem.

Finally, the detailed investigation of more complex dependencies and database modifications in the form of updates and insertions offer possibilities for further research.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, London (1995)
2. Bayer, R., McCreight, E.M.: Organization and maintenance of large ordered indices. Acta Informatica 1(3), 173–189 (1972)
3. Bell, D.E., LaPadula, L.J.: Secure computer systems: Unified exposition and MULTICS interpretation. Technical Report ESD-TR-75-306, The MITRE Corporation (1975)
4. Bertino, E., Sandhu, R.S.: Database security—concepts, approaches, and challenges. IEEE Trans. Dependable Sec. Comput. 2(1), 2–18 (2005)
5. Bishop, M.: Computer Security: Art and Science. Addison-Wesley, London (2003)
6. Biskup, J.: For unknown secrecies refusal is better than lying. Data Knowl. Eng. 33(1), 1–23 (2000)
7. Biskup, J., Bonatti, P.A.: Lying versus refusal for known potential secrets. Data Knowl. Eng. 38(2), 199–222 (2001)
8. Biskup, J., Bonatti, P.A.: Confidentiality policies and their enforcement for controlled query evaluation. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ES-ORICS 2002. LNCS, vol. 2502, pp. 39–54. Springer, Heidelberg (2002)
9. Biskup, J., Bonatti, P.A.: Controlled query evaluation for known policies by combining lying and refusal. Ann. Math. Artif. Intell. 40(1-2), 37–62 (2004)
10. Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Sec. 3(1), 14–27 (2004)
11. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 43–62. Springer, Heidelberg (2006)

12. Bonatti, P.A., Kraus, S., Subrahmanian, V.S.: Foundations of secure deductive databases. IEEE Trans. Knowl. Data Eng. 7(3), 406–422 (1995)
13. Brodsky, A., Farkas, C., Jajodia, S.: Secure databases: Constraints, inference channels, and monitoring disclosures. IEEE Trans. Knowl. Data Eng. 12(6), 900–919 (2000)
14. Cuppens, F., Gabillon, A.: Logical foundations of multilevel databases. Data Knowl. Eng. 29(3), 259–291 (1999)
15. Cuppens, F., Gabillon, A.: Cover story management. Data Knowl. Eng. 37(2), 177–201 (2001)
16. Dawson, S., di Vimercati, S.D.C., Samarati, P.: Specification and enforcement of classification and inference constraints. In: IEEE Symposium on Security and Privacy, pp. 181–195. IEEE Computer Society Press, Los Alamitos (1999)
17. Dawson, S., di Vimercati, S.D.C., Lincoln, P., Samarati, P.: Minimal data upgrading to prevent inference and association. In: Proc. PODS 1999, pp. 114–125 (1999)
18. Denning, D.E.: Cryptography and Data Security. Addison-Wesley, London (1983)
19. Farkas, C., Jajodia, S.: The inference problem: A survey. SIGKDD Explorations 4(2), 6–11 (2002)
20. Fernández-Medina, E., Piattini, M.: Designing secure databases. Information & Software Technology 47(7), 463–477 (2005)
21. Gollmann, D.: Computer Security, 2nd edn. John Wiley & Sons, Chichester (2006)
22. Lunt, T.F., Denning, D.D., Schell, R.R., Heckman, M., Shockley, W.R.: The SeaView security model. IEEE Trans. Software Eng. 16(6), 593–607 (1990)
23. McLean, J.: A comment on the 'Basic Security Theorem' of Bell and LaPadula. Inf. Process. Lett. 20(2), 67–70 (1985)
24. McLean, J.: Reasoning about security models. In: IEEE Symposium on Security and Privacy, pp. 123–131. IEEE Computer Society Press, Los Alamitos (1987)
25. McLean, J.: The specification and modeling of computer security. IEEE Computer 23(1), 9–16 (1990)
26. Nicomette, V., Deswarte, Y.: A multilevel security model for distributed object systems. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 80–98. Springer, Heidelberg (1996)
27. Olivier, M.S., von Solms, S.H.: A taxonomy for secure object-oriented databases. ACM Trans. Database Syst. 19(1), 3–46 (1994)
28. Rizvi, S., Mendelzon, A., Sudarshan, S., Roy, P.: Extending query rewriting techniques for fine-grained access control. In: Proc. ACM SIGMOD 2004, pp. 551–562. ACM Press, New York (2004)
29. Rosenthal, A., Sciore, E., Wright, R.N.: Simplifying policy administration for distributed privacy-preserving computation (2007), http://www.mitre.org/staffpages/arnie/pubs/simplifying_policy_admin_privacy_reserving.pdf
30. Sandhu, R.: Lattice-based access control models. IEEE Computer 26(11), 9–19 (1993)
31. Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering queries without revealing secrets. ACM Trans. Database Syst. 8(1), 41–59 (1983)
32. Stonebraker, M., Wong, E.: Access control in a relational data base management system by query modification. In: Proc. ACM/CSC-ER Annual Conference, pp. 180–186 (1974)

Appendix: Detailed Proofs

Lemma 1 and Lemma 2

Suppose that $dq(\chi)$ denotes the formula emerging from $\chi \in \mathscr{L}_{\mathrm{ps}}$ by discarding the quantifiers. E. g., if $\chi = (\exists X)R(a, X)$, then $dq(\chi) = R(a, X)$.

Proof (Lemma 1). First, we show " $\Longrightarrow$ " by contraposition. Assume, for every $\chi_i \in \{\chi_1, \ldots, \chi_p\}$ it holds that χ is not relevant for χ_i, leading to

$$\beta(dq(\chi_i)) \neq \beta(dq(\chi)) \tag{1}$$

for every variable assignment β. We construct a witness instance r against condition 1 by taking any variable assignment β and setting

$$r := \{\beta(dq(\chi_1)), \ldots, \beta(dq(\chi_p))\}\,. \tag{2}$$

Then, $r \models_{\mathrm{M}} \{\chi_1, \ldots, \chi_p\}$ by (2) and $r \not\models_{\mathrm{M}} \chi$ by (1) and (2).

Second, we show " $\Longleftarrow$ ". Let there be a χ_i that χ is relevant for. By definition of the existential quantifier, it holds that

$$\begin{aligned}&(\exists X_1)\ldots(\exists X_l)R(v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_n) \models\\ &(\exists X_1)\ldots(\exists X_l)(\exists X_{l+1})R(v_1, \ldots, v_{i-1}, X_{l+1}, v_{i+1}, \ldots, v_n)\end{aligned} \tag{3}$$

with R denoting a relation symbol, $v_i \in \mathit{Const}$, $X_{l+1} \in \mathit{Var}$, and $X_{l+1} \notin \{X_1, \ldots, X_l\}$. Equation (3) in combination with Definition 10 yields $\chi_i \models \chi$ and thus $\{\chi_1, \ldots, \chi_p\} \models \chi$ as well. □

Proof (Lemma 2). Again, we first show " $\Longrightarrow$ " by contraposition and assume that for every $\chi \in \mathscr{L}_{\mathrm{q}}$ it holds that

$$\chi \notin \mathcal{S} \text{ or } \chi \not\models \Psi. \tag{4}$$

We construct a witness instance r against condition 1, satisfying $r \models_{\mathrm{M}} \mathcal{S}$ and $r \not\models_{\mathrm{M}} \Psi$. Consider r as the Herbrand interpretation given by

$$r := \mathcal{S}_1. \tag{5}$$

Since $\mathcal{S}$ is finite, r can be represented as a finite relation as well. It holds that

$$r \models_{\mathrm{M}} \mathcal{S}, \tag{6}$$

as we show by distinguishing the three kinds of formulas in $\mathcal{S}$:

1. [$\mathcal{S}_1$] By the construction of r according to (5) it holds that $r \models_{\mathrm{M}} \mathcal{S}_1$.
2. [$\mathcal{S}_2$] By the consistence of $\mathcal{S}$, there is no formula in $\mathcal{S}_1$ whose negative complement is in $\mathcal{S}_2$. Thus, by the construction of r and the closed world assumption, $r \models_{\mathrm{M}} \mathcal{S}_2$.

3. [$\mathcal{S}_3$] Let $\mathcal{S}_1$ consist of the tuples $\mu_1, \ldots, \mu_k$. From (5) it follows that $r \models_{\mathrm{M}} \mu_1, r \models_{\mathrm{M}} \mu_2, \ldots, r \models_{\mathrm{M}} \mu_k$. Now assume a $\phi \in \mathcal{S}_3$ with $r \not\models_{\mathrm{M}} \phi$, meaning that the fd $\mathcal{A} \rightarrow \mathcal{B}$ represented by ϕ is violated in r. As a result, two tuples μ_i and μ_j from $\mathcal{S}_1$ agreeing on the $\mathcal{A}$-attributes must differ on the $\mathcal{B}$-attributes. However, this implies that $\{\mu_i, \mu_j, \phi\}$ is inconsistent and, because of $\{\mu_i, \mu_j, \phi\} \subseteq \mathcal{S}$, that $\mathcal{S}$ is inconsistent as well, contradicting the precondition that $\mathcal{S}$ is a consistent set.

From (4) we know that $\phi \not\models \Psi$ for every $\phi \in \mathcal{S}_1$. Since $\mathcal{S}_1$ is consistent and does not contain any functional dependencies, there are no further inferences possible, leading to $\mathcal{S}_1 \not\models \Psi$. Then, by Lemma 1 and Definition 10, it follows for Ψ that for every $\phi \in \mathcal{S}_1$ there exists an $A \in \mathcal{U}$ with $\Psi(A) \in \mathit{Const}$ and $\phi(A) \neq \Psi(A)$. With (5) this leads to $r \not\models_{\mathrm{M}} \Psi$.

Second, we examine the "$\Longleftarrow$" case. Obviously, by $\chi \models \Psi$ and $\chi \in \mathcal{S}$, it holds that $\mathcal{S} \models \Psi$. Consequently, the proof is complete. □

Theorem 1

Proof. Consider a finite prefix $Q' = \langle \Phi_1, \ldots, \Phi_n \rangle$ of Q and a potential secret Ψ from *pot_sec*. We prove the existence of an instance r' making Σ and the answers to the allowed queries in Q' true on the one hand and making Ψ false on the other hand, thereby being an instance of RS and satisfying the properties of Definition 7. To this end, we construct an auxiliary set *log*:

$$\mathit{log} := \Sigma \cup \{\mathit{ans}_i \mid i \in \{1, \ldots, n\}, \mathit{ans}_i \neq \texttt{mum}\}. \tag{7}$$

We now show indirectly that *log* does not imply Ψ after the last query and therefore assume

$$\mathit{log} \models \Psi. \tag{8}$$

By Lemma 2 (with $\mathcal{S} := \mathit{log}$), (8) would require a positive $\mathit{ans}_i \in \mathit{log}$, satisfying

$$\mathit{ans}_i \models \Psi. \tag{9}$$

According to (7) and Definition 12, $\mathit{ans}_i = \mathit{eval}^*(\Phi_i)(r)$, and thus, by (9),

$$\mathit{eval}^*(\Phi_i)(r) \models \Psi. \tag{10}$$

By Definition 4, $\mathit{eval}^*(\Phi_i)(r) \in \{\Phi_i, \neg\Phi_i\}$, so we actually have to distinguish two cases in (10). However, by the structure of Φ_i and Ψ and by the infiniteness of *Const*, $\neg\Phi_i \models \Psi$ cannot hold; consequently, (10) reduces to

$$\Phi_i \models \Psi. \tag{11}$$

According to Definition 9, s contains a tuple μ such that Ψ is relevant for μ and every attribute in μ not agreeing with Ψ has the value #. Furthermore, Ψ is relevant for Φ_i by (11). This leads to Φ_i not being allowed w. r. t. *pot_sec*, since μ

contradicts the condition in Definition 11. As a result, $ans_i = \texttt{mum}$, contradicting (9) and thereby (8). Consequently,

$$log \not\models \Psi. \tag{12}$$

Now, we can prove ac secure w. r. t. pot_sec: By (12) and the definition of logical implication, there exists a database instance r' satisfying

$$r' \models_{\mathrm{M}} log \text{ and } r' \not\models_{\mathrm{M}} \Psi. \tag{13}$$

Equation (7) implies $log \models ans_i$ for every $ans_i \neq \texttt{mum}$, which, in combination with (13), leads to

$$r' \models_{\mathrm{M}} ans_i \text{ for every } ans_i \neq \texttt{mum}. \tag{14}$$

Since access control in the sense of Definition 12 is instance independent, the i-th answer of ac only depends on Φ_i and pot_sec. As a result,

$$ac(\Phi_i)(r, pot\_sec) = \texttt{mum} \quad \text{iff} \quad ac(\Phi_i)(r', pot\_sec) = \texttt{mum}. \tag{15}$$

Access control either returns the correct answer to a query or refuses the answer. Consequently, (14) and (15) imply the first property of Definition 7, $ac(Q')(r, pot\_sec) = ac(Q')(r', pot\_sec)$. By (13), also the second property, $eval^*(\Psi)(r') = \neg\Psi$, is satisfied. Finally, by (13) and (7) r' satisfies Σ and is therefore an instance of RS. □

Efficient Negative Databases from Cryptographic Hash Functions

George Danezis, Claudia Diaz, Sebastian Faust, Emilia Käsper,
Carmela Troncoso, and Bart Preneel

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

Abstract. A negative database is a privacy-preserving storage system that allows to efficiently test if an entry is present, but makes it hard to enumerate all encoded entries. We improve significantly over previous work presented at ISC 2006 by Esponda *et al.* [9], by showing constructions for negative databases reducible to the security of well understood primitives, such as cryptographic hash functions or the hardness of the Discrete-Logarithm problem. Our constructions require only $\mathcal{O}(m)$ storage in the number m of entries in the database, and linear query time (compared to $\mathcal{O}(l \cdot m)$ storage and $\mathcal{O}(l \cdot m)$ query time, where l is a security parameter.) Our claims are supported by both proofs of security and experimental performance measurements.

1 Introduction

In a celebrated series of academic papers [8,9], which have also attracted wider public interest [1], Fernando Esponda *et al.* introduce the concept of *negative databases* to protect the privacy of stored records. A negative database is a representation of a set of records (the positive database) that allows its holder to test whether particular entries are present in the database, but makes it very hard to efficiently enumerate all entries.

Negative databases have a wide range of applications with the potential to enhance privacy: holders of records cannot easily retrieve all entries, and lost or compromised machines do not therefore lead to large scale privacy compromises. As a result, data holders can also share information without fear that the receiver will be able to extract the full contents of the database. As an example the Transport Security Administration can provide a black list of passengers as a negative database to airline companies, who can use it to check whether passengers to fly with them are on the list. Yet the companies would not be able to extract the full contents of the database unless they can perform an exhaustive search on it.

The protocols of Esponda *et al.* for building negative databases are limited in many ways, and lead to a database of size $\mathcal{O}(l \cdot m)$ entries – where m is the number of entries in the positive database, and l the size of each entry in bits (with the

J. Garay et al. (Eds.): ISC 2007, LNCS 4779, pp. 423–436, 2007.

restriction that $l > 1000$ for security, thus leading to total storage requirement of $\mathcal{O}(l^2 \cdot m)$ bits.) Furthermore, the security of their scheme relies on generating and not being able to solve hard instances of the 3-SAT problem, which is a non-standard security assumption in computer security and cryptography (but note that the 3-SAT problem is NP-complete). For a detailed security analysis the reader is referred the original papers.

In this work we present two constructions that provide exactly the same functionality as the original negative database schemes. Our constructions are computationally efficient for all operations and lead to much more compact "negative" representations. We prove the security of our constructions using standard cryptographic reductions to the security of well understood primitives, such as cryptographic hash functions for our first scheme and the family of Discrete-Logarithm (DL) assumptions for the second scheme. Experimental results demonstrate that implementing our first scheme is straightforward and can efficiently scale to databases of many megabytes.

The rest of the paper is organised as follows: Section 2 presents the background and related work, including a brief overview of previous designs for negative databases. Section 3 presents our schemes, based on cryptographic hash functions and the DL assumptions, with security proofs being presented in Sect. 4. We evaluate the theoretical and experimental efficiency of the schemes in Sect. 5. Further privacy enhancing extensions, intrinsic limitations and conclusions are presented in Sections 6, 7, and 8 respectively.

2 Related Work

The concept of negative databases was introduced by Esponda *et al.* [8,9]. Each entry in the database is represented as a bit-string of length l and the set of all bit strings *not in the database* is represented in a compact form. The compact form is a set of l bit-strings of length l each composed of '0's, '1's and wild cards that could match either (i.e. '*'s). To test whether a string is present in the "positive" database, one checks whether it matches any of the negative entries: the string is in the positive database if it does not match any of the negative representations.

The security of the scheme is based on the inability of the adversary to infer which positive strings are in the database in a way that is more efficient than enumerating all possible strings and testing them one by one. Esponda *et al.* reduce the complexity of breaking the security of their proposal to solving a hard instance of a 3-SAT problem; they conjecture that the problem becomes intractable for strings of $l > 1000$ bits. Their scheme does not allow for multiple entries to be encoded in a combined way, and therefore they require a negative database of size $\mathcal{O}(l)$ to be generated for each of the m positive entries, leading to a storage requirement of $\mathcal{O}(l^2 \cdot m)$ bits. Attempts to represent more than one positive entry in an integrated manner lead to shortcuts in solving the 3-SAT problem on which the security of the scheme is based. Dummy entries representing no string have to be included to obfuscate the size of the positive

database, and check sums (CRC or MD5 are proposed) are appended to the positive entries to minimise false positives.

Once constructed, negative databases can be used to efficiently check whether a particular string, or part of a string, is present in the positive database. The need to have check sums appended to the entries (to avoid false positives) restricts the ability to query partial contents, and only allows to test whether the full contents of a particular field are present in some row.

Our constructions for negative databases support the same operations, and we reduce their security to the security of well known cryptographic primitives, such as cryptographic hash functions. The idea of applying hash functions in the context of protecting weakly chosen password databases has been proposed before by Needham *et al.* [12]. It is now widely adopted to protect the confidentiality of password files against accidental disclosure or corrupt insiders on most UNIX systems. Our work starts from this idea but generalises it to database tables with arbitrary numbers of columns (or fields) per row. We allow complex queries on such databases, as well as merging databases, and adding and deleting entries. Our constructions based on discrete-log and related assumptions further allow any party to prove properties of the entries in the negative database without revealing any information.

3 Our Schemes

Our goal is to efficiently implement the same functionalities as negative databases in [10], with cryptographic security guarantees. In order to achieve this, our schemes should satisfy the following properties, described in [10]:

- **Hard to reverse.** Given a negative database NDB, there should be no algorithm for obtaining the positive image DB that is more efficient than exhaustive search.
- **Singleton negative database.** Each hard-to-reverse entry in NDB represents either a string in DB, or no string at all, i.e., reversing the database does not introduce "false" positive entries.
- **Easy to update.** There should be efficient algorithms for adding and deleting entries from DB.
- **Obfuscated size.** The size of the positive image DB should not be visible from NDB.
- **Probabilistic.** A particular binary string $s \in \mathsf{DB}$ should have many possible representations in NDB.

Esponda *et al.* mention an additional property [10]:

> "**String based:** One of the more salient features of our scheme is that it is based on string matching. This permits us to meaningfully affect a positive image by manipulating the entries of its negative database; references [ref17,ref18] discuss some applications of this idea. In the coming paragraphs we present an operation that illustrates the usefulness of this property."

Algorithm 1. Generating a hard-to-reverse negative database

INPUT: Database $\mathsf{DB} = \{\mathsf{DB}_{i,j}\}$, $\mathsf{DB}_{i,j} \in \mathcal{M}$, $i = 0, \ldots, m-1$, $j = 0, \ldots, n-1$
Function $H : \mathcal{R} \times \mathcal{M} \mapsto \mathcal{T}$
OUTPUT: Negative database $\mathsf{NDB} = \{\mathsf{NDB}_{i,j}\}$, $\mathsf{NDB}_{i,j} \in \mathcal{R} \times \mathcal{T}$
1: Initialize $\mathsf{NDB} = \{\}$
2: **for** $i = 0$ to $m-1$ **do**
3: **for** $j = 0$ to $n-1$ **do**
4: Randomly choose $r_{i,j} \in_R \mathcal{R}$
5: Compute $\mathsf{NDB}_{i,j} = (r_{i,j}, H(r_{i,j}, \mathsf{DB}_{i,j}))$
6: Set $\mathsf{NDB} = \mathsf{NDB} \cup \{\mathsf{NDB}_{i,j}\}$

Algorithm 2. Obfuscating database size

INPUT: Negative database $\mathsf{NDB} = \{\mathsf{NDB}_{i,j}\}$, $\mathsf{NDB}_{i,j} \in \mathcal{R} \times \mathcal{T}$, $i = 0, \ldots, m-1$, $j = 0, \ldots, n-1$
integer $d > 0$
OUTPUT: Negative database $\mathsf{NDB}' \supset \mathsf{NDB}$ with d dummy entries
1: Initialize $\mathsf{NDB}' = \mathsf{NDB}$
2: **for** $i = m$ to $m + d - 1$ **do**
3: **for** $j = 0$ to $n-1$ **do**
4: Randomly choose $r_{i,j} \in_R \mathcal{R}$, $t_{i,j} \in_R \mathcal{T}$
5: Set $\mathsf{NDB}'_{i,j} = (r_{i,j}, t_{i,j})$
6: Set $\mathsf{NDB}' = \mathsf{NDB}' \cup \{\mathsf{NDB}'_{i,j}\}$

Our conversion of DB elements to their negative form involves transformations that destroy their semantic structure, as opposed to the string based approach in [10]. Nevertheless, after thorough examination of the examples given in [10] we have not found any property or functionality provided by the string based feature that our schemes cannot satisfy. More details on functionalities can be found in Sect. 6.1.

3.1 Algorithms for Creating, Updating and Searching in NDB

Our negative database construction is based on a one-way function, for which we propose two alternative implementations: the first is based on cryptographic hash functions (explained in Sect. 3.3), and the second on the family of discrete log assumptions (Sect. 3.4). For now, we make an abstraction of the one-way function and present general algorithms for generating the negative database, obfuscating its size, and verifying if a string s is in the database.

Let DB be a database that contains m records with n fields each; i.e., a total of $m \cdot n$ elements. We denote by $\mathsf{DB}_{i,j}$ the contents of the element (i, j) corresponding to the j-th field of the i-th record in DB, and $\mathcal{M}$ the universe of all possible element contents.

From DB, we can efficiently generate NDB following the algorithm shown in Alg. 1. For each element $\mathsf{DB}_{i,j} \in \mathcal{M}$, we generate a random number $r_{i,j} \in \mathcal{R}$. We

Algorithm 3. Verifying "is s in DB"?

INPUT: Negative database $\mathsf{NDB} = \{\mathsf{NDB}_{i,j}\}$ corresponding to DB, $\mathsf{NDB}_{i,j} \in \mathcal{R} \times \mathcal{T}$, $i = 0, \ldots, m-1$, $j = 0, \ldots, n-1$
index $k \in \{0, \ldots, n-1\}$, value $s \in \mathcal{M}$
OUTPUT: Verify if $s = \mathsf{DB}_{i,k}$ for some index i

```
1: for i = 0 to m − 1 do
2:     Let NDB_{i,k} = (r_{i,k}, t_{i,k})
3:     if H(r_{i,k}, s) = t_{i,k} then
4:         return true
5: return false
```

define a suitable one-way function $H : \mathcal{R} \times \mathcal{M} \mapsto \mathcal{T}$ that maps a pair of values $(r_{i,j}, \mathsf{DB}_{i,j})$ to an element $t_{i,j} \in \mathcal{T}$, i.e., $t_{i,j} = H(r_{i,j}, \mathsf{DB}_{i,j})$. The element $\mathsf{NDB}_{i,j}$ in position (i, j) of the negative database is the tuple $(r_{i,j}, t_{i,j})$. The randomized representation of element $\mathsf{DB}_{i,j}$ in NDB provides additional security guarantees (see Sect. 4).

In order to obfuscate the number m of records in DB, we add d dummy entries to NDB, as shown in Alg. 2. The dummy entries are pairs $(r_{i,j}, t_{i,j})$ of random elements from $\mathcal{R} \times \mathcal{T}$. We note that $m+d$ gives an upper bound on the real size of the database; i.e., DB would be known to contain a maximum of $m + d$ real records.

Querying NDB in order to check if an element s is in the database is done following Alg. 3. Normally, the user querying the database would want to know if an element s is present in a given field (e.g., "is name s contained in the names column?"). In order to do the query, the user provides the string s and the column k (field) where s is expected to appear. Then, the function H is applied to all pairs $(r_{i,k}, s)$ to check if the result matches the tag $t_{i,k}$ for some record i. If there is a match, then we confirm that s is in DB, otherwise the answer is negative.

3.2 Properties of Our System

Although our NDB is not constructed by "negating" DB, we can show that it has the same functionalities and properties as the negative databases described in [10]. First, as NDB is constructed using a one-way function, obtaining DB from NDB is a hard problem. We provide security arguments and proofs of this property in Sect. 4. From the algorithms for constructing NDB and obfuscating the size of DB, we can see that (hard-to-reverse) entries in NDB represent either a string s in DB (if they have been generated by applying the one-way function to s), or a random dummy string (if they have been randomly generated for obfuscating the size of DB). The size of the positive image corresponding to some NDB is obfuscated, since it is hard to distinguish dummy entries from those that correspond to an element in DB (as proven in Sect. 4). The size of NDB reveals only an upper bound to the size of DB.

It is very easy to update DB by adding and deleting entries from NDB. It is sufficient to apply the one way function to the entry, and then add (delete) its negative image to (from) NDB. Our scheme is probabilistic, as a particular

binary string s has many possible negative database representations (as many as possible values for the random $r_{i,j}$), and the creation process chooses one uniformly at random. Given two negative database entries, it is hard to determine if they represent the same value. We prove this property formally in Thm. 3.

3.3 Negative Databases from Cryptographic Hash Functions

Cryptographic hash functions such as SHA-256 [14,16] and RIPEMD-160 [7,14] are widely used cryptographic primitives. They are compressing functions that take a variable size input and return a fixed size output (of 256 and 160 bits, respectively). The key properties of cryptographic hash functions are preimage and second preimage resistance and collision resistance. Loosely speaking, these mean that given a hash value $h(x)$ it is difficult to find x; given $x, h(x)$ it is difficult to find another y such that $h(y) = h(x)$; and it is difficult to find arbitrary x, y such that $h(y) = h(x)$.

Extensive cryptographic research has gone into understanding the security of hash functions, with spectacular results demonstrating the insecurity [19] of the standard MD5 [17] and SHA-1 [14,16] algorithms. The weaknesses concentrate on the general collision resistance of these functions which is not required to show the security of our designs, but it is still prudent to migrate to the use of functions such as SHA-256 and RIPEMD-160 that are still believed to be secure under all security notions.

Thus, we instantiate the one-way function H with a cryptographic hash-function h, so that $H(r_{i,j}, \mathsf{DB}_{i,j}) = h(r_{i,j}||\mathsf{DB}_{i,j})$. In this particular application, the adversary knows $r_{i,j}$, hence partial preimage resistance is required to guarantee the hard-to-reverse property. One can prove even stronger properties in the random oracle model [2]. In this model, practical hash functions are modelled in an idealized way, that is, as "black-box" functions that output a uniformly random value as response to every new query. If the random oracle is queried again with the same input, it outputs the same value as before.

3.4 Negative Databases Based on Discrete-Log and DDH Assumptions

We propose a second instantiation for the one-way function, this time based on the hardness of the discrete logarithm problem. Namely, let p be a large prime and $G =< g >$ a multiplicative group of prime order p. Let $\mathbb{Z}_p$ be the additive group of integers modulo p. Then, we set $H : G \setminus \{1\} \times \mathbb{Z}_p \mapsto G$ as $H(g^r, m) = g^{rm}$.

To reason formally about the security of our scheme we have to introduce the notation of negligibility.

Negligibility: *A function f is* negligible *if for every polynomial $P(k)$, $f(k) \leq \frac{1}{|P(k)|}$ for all sufficiently large k.*

The security of our scheme relies on the discrete-log assumption (where the security parameter k is the bit-length of p):

Discrete-Logarithm (DL) Assumption: *For every probabilistic polynomial time algorithm A, the probability* $Pr[g \leftarrow G; x \leftarrow \mathbb{Z}_p; A(p, g, g^x) = x]$ *is negligible.*

If the DL assumption holds for a group G, then $f(x) = g^x$ is a one-way function. Thus, H can indeed be instantiated with $H(g^r, m) = g^{rm}$. Furthermore, in Sect. 4, we prove additional properties of the NDB under the DDH assumption:

Decisional-Diffie-Hellman (DDH) Assumption: *For every probabilistic polynomial time algorithm A, the probability*

$$|Pr[g \leftarrow G; x, y \leftarrow \mathbb{Z}_p; A(p, g, g^x, g^y, g^{xy}) = 1] - Pr[g \leftarrow G; x, y, z \leftarrow \mathbb{Z}_p; A(p, g, g^x, g^y, g^z) = 1]|$$

is negligible.

The DDH assumption says that given a tuple (g^x, g^y, g^z), it is computationally infeasible to decide whether $z = xy$. Evidently, the DDH assumption implies the DL assumption.

4 Security Arguments and Proofs

We start by proving that a user can indeed query single values in the negative database, i.e., that Alg. 3 returns the wrong answer with negligible probability (in the length of the entries in NDB). In the first construction, the negligible error probability cannot be avoided if one wants to use a compressing hash function instead of a bijective function. In the second case, the negligible error is introduced by dummy entries; if the database size is not obfuscated, the algorithm is always correct.

Theorem 1. *Assume that $H : \mathcal{R} \times \mathcal{M} \mapsto \mathcal{T}$ is a random oracle. Then Alg. 3 returns the correct answer with probability at least $1 - \frac{m+d}{|\mathcal{T}|}$, where m and d are the number of real and dummy records (rows) in the database, respectively.*

Proof. If $s = \mathsf{DB}_{i,k}$ is in the database, then the algorithm clearly returns the correct answer. Assume now that $s \in \mathcal{M}$ is not in the database. For any entry $\mathsf{NDB}_{i,k} = (r_{i,k}, t_{i,k})$, the algorithm incorrectly returns `true` if and only if $H(r_{i,k}, s) = t_{i,k}$. Since the answers of the random oracle are uniformly distributed, $\Pr[H(r_{i,k}, s) = t_{i,k}] = \frac{1}{|\mathcal{T}|}$, and the result follows from the union bound. □

Theorem 2. *Let $G = < g >$ be a multiplicative group of prime order $|G| = p$ generated by g. Define the function $H : G \setminus \{1\} \times \mathbb{Z}_p \mapsto G$ as $H(g^r, m) = g^{rm}$. Then Alg. 3 returns the correct answer with probability at least $1 - \frac{d}{p}$, where d is the number of dummy entries in the database.*

Proof. If $s = \mathsf{DB}_{i,k}$ is in the database, then the algorithm clearly returns the correct answer. Assume now that $s \in \mathbb{Z}_p$ is not in the database. For any "real"

entry $\mathsf{DB}_{i,k} = s' \neq s \pmod p$ and corresponding negative entry $\mathsf{NDB}_{i,k} = (g^{r_{i,k}}, g^{r_{i,k}s'})$, the algorithm correctly computes $H(g^{r_{i,k}}, s) = g^{r_{i,k}s} \neq g^{r_{i,k}s'}$. For any "dummy" entry $\mathsf{NDB}_{i,k} = (g^{r_{i,k}}, g^{t_{i,k}})$, the algorithm incorrectly returns `true` if and only if $H(g^{r_{i,k}}, s) = g^{t_{i,k}}$. Since $g^{t_{i,k}}$ was drawn uniformly at random from G, $\Pr[H(g^{r_{i,k}}, s) = g^{t_{i,k}}] = \frac{1}{p}$. The result then follows from the union bound. □

Next, we turn our attention to privacy-preserving properties. Since our negative databases are constructed using one-way functions, obtaining the positive representation from the negative one implies breaking the one-wayness assumption. In particular, the security of our two constructions relies on the (partial) preimage resistance of hash functions in the first case, and the hardness of discrete logarithm in groups of prime order in the second case.

A standard design goal for hash functions is that they should withstand preimage attacks faster than exhaustive search. More precisely, let $l = \log_2 |\mathcal{M}|$ be the length of each entry in the positive database and let $r = \log_2 |\mathcal{R}|$ be the length of random values. Since values $r_{i,j}$ are known to the adversary, exhaustive search for inverting a fixed value in NDB takes 2^l steps in the worst case (assuming that all l-bit strings are possible database entries). Inverting the whole NDB takes at most $mn2^l$ steps. However, since the values $r_{i,j}$ are chosen randomly and only become public when the database is published, full precomputation *before seeing* NDB takes 2^{l+r} steps. Concretely, we propose to use random values $r_{i,j}$ of 128–256 bits to thwart offline attacks. Choosing $r \geq 128$ is also sufficient to guarantee that the random values never repeat: by the Birthday Paradox, collisions only become likely after $\mathcal{O}(2^{r/2})$ choices.

Moreover, under stronger but still reasonable assumptions, our constructions benefit from *indistinguishability* of entries: given two negative database entries $\mathsf{NDB}_{i,j}$ and $\mathsf{NDB}_{i',j'}$, a polynomial-time adversary cannot decide with non-negligible probability whether these entries correspond to the same value in the positive database, i.e, whether $\mathsf{DB}_{i,j} = \mathsf{DB}_{i',j'}$. For the first construction, the result is obvious if we substitute the hash function with a random oracle and assume that random values $r_{i,j}$ never repeat. We now prove the result for the second case with a tight reduction to the Decisional Diffie-Hellman assumption.

Theorem 3. *Let $G = <g>$ be a multiplicative group of prime order $|G| = p$ generated by g. Define the function $H : G \setminus \{1\} \times \mathbb{Z}_p \mapsto G$ as $H(g^r, m) = g^{rm}$. Given two* NDB *entries $\mathsf{NDB}_{i,j} = (g^r, g^x)$, $\mathsf{NDB}_{i',j'} = (g^{r'}, g^{x'})$, it is computationally infeasible to decide whether $\mathsf{DB}_{i,j} = \mathsf{DB}_{i',j'}$ ($\log_{g^r} g^x = \log_{g^{r'}} g^{x'}$) under the DDH assumption.*

Proof. Given a probabilistic polynomial-time adversary $\mathcal{A}$ that can distinguish between database entries with advantage ε, we construct another adversary $\mathcal{B}$ that can break the DDH assumption with advantage ε.

Let (g^x, g^y, g^z) be the challenge DDH tuple given to $\mathcal{B}$. We let $\mathcal{B}$ randomly choose $r \in \mathbb{Z}_p$, construct tuples (g^r, g^{xr}) and (g^{y-r}, g^{z-xr}) and send them to $\mathcal{A}$. It is easy to see that $\mathcal{B}$ can solve the DDH problem with advantage ε, since:

$$\log_{g^r} g^{xr} = \log_{g^{y-r}} g^{z-xr} \;\Leftrightarrow\; x(y-r) = z - xr \;\Leftrightarrow\; xy = z. \qquad \square$$

If the adversary has some *a priori* information about the entry m, fast attacks such as the baby-step giant-step method for finding the discrete logarithm may however be possible. We discuss the impact of field entropy on security in Sect. 7.

5 Evaluation

5.1 Efficiency

In this section we discuss the efficiency of our approach, and compare it to [9], both in terms of space and time complexity.

In the scheme proposed by Esponda *et al.* [9], positive database (DB) entries cannot be negatively represented in a global way. Instead, each entry has to be represented individually by a negative database ($\mathsf{NDB}_{i,j}$). Each of these $\mathsf{NDB}_{i,j}$ has size $\mathcal{O}(l^2)$, where l is the size of the entry in DB ($l > 1000$ for security reasons), as their scheme needs l entries of l bits per entry in the NDB in order to conceal the positive representation. Assuming that DB contains m entries, the complete NDB will occupy $\mathcal{O}(l^2 \cdot m)$ bits of space.

Our construction, on the contrary, stores only one value per entry in DB, such that the global size of the final NDB is linear in the size of the original database, occupying $\mathcal{O}((t+r) \cdot m)$ bits for a positive database with m entries, where $t = \log_2 |T|$ is the length of the output of the one-way function H and $r = \log_2 |R|$ is the length of the random value. This may even lead to a negative database that is smaller than the original positive database (when positive entries are larger than $t + r$).

In terms of time complexity, our proposal is more efficient than the original scheme. For answering the query "Is q in DB?", the original approach needs to check every value in every NDB. This takes $\mathcal{O}(l \cdot m)$ time. In our approach, only m entries need to be checked. Thus, the query response time is linear in the size of the database, $\mathcal{O}(t_H \cdot m)$, depending on the time needed to execute the one-way function, t_H.

Our calculations so far concern entries with a single field. When entries have multiple fields n, the space complexity increases linearly by the same factor n. Query time complexity does not increase, provided that the query is restricted to a single field.

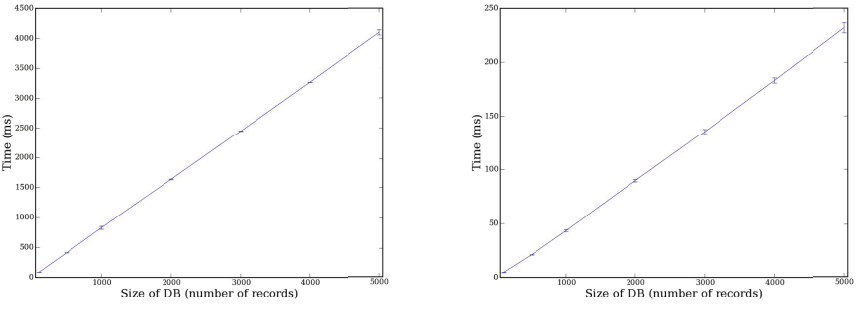
Cryptographic hash functions such as the SHA family are very fast on commodity processors, and can achieve speeds of up to 1 Gb/s in dedicated hardware [15]. Modular exponentiation is more expensive on commodity hardware, but techniques based on the precomputation of some values provide a considerable speed-up [3,13]. Specialized hardware achieves a rate of about 50000 exponentiations per second [18].

5.2 Experimental Results

We have implemented a simulation in Python in order to test the efficiency of our proposal. In our example, each entry of the database consists of six fields: name, family name, gender, credit card number, month of expiration and year of expiration, as shown in Table 1. All of these fields were of length 20 bytes whereas normally gender, month, year, etc length should be smaller. This gives us an upper bound on performance (but should have little effect in practice.) We use SHA-1 [16] as the one-way function to create the negative image of the database[1], with 20-byte values to randomize the output (or $r = 160$).

Table 1. Database entries

Name	Family name	Gender	Credit Card Number	Month of Expiration	Year of Expiration



(a) Time for create a database by size (b) Response time by size of the database

Fig. 1. Simulation results on a 1 GHz Pentium M

First, we tested the time of creation of a database depending on the number of entries. As shown in Fig. 1(a), the time for creating a database is linear in its size, and smaller than 5 seconds for a 5000-entry database (with seven 20-byte fields each).

Our second test measured the response time of our scheme. We assume the worst case, when the query q is not in DB, thus, all of the entries need to be checked to give a final negative response. As expected, the time is linear with the number of entries in the database (see Fig. 1(b)).

[1] SHA-1 was chosen for ease of implementation, since it is available in the standard libraries. Even if current shortcut attacks only reduce the complexity to find collisions, SHA-1 should not be used in production systems. Instead, we recommend RIPEMD-160 or SHA-256 that have comparable performance characteristics.

6 Extensions and Discussion

6.1 Intersection of Databases

One of the advantages of negative databases is that they enable organizations to compare their negative database images without jeopardizing the privacy of the data subjects, or leaking sensitive information to company outsiders. Our schemes, in spite of destroying the semantics of the original (positive) database, support these operations.

For example, users can do private "select and project" operations without first transmitting the whole NDB. Namely, a user interested in entries that have a certain value v in field (column) f can request the entries in column f, apply the one-way function locally to v (salted appropriately with the random values), and send the indices of matching entries to the owner of NDB. The latter can then create and send back a new NDB' of matching entries without learning the search criterion v. Following the example in [10], authorities can also take an intersection of two positive DBs without revealing their interests to the database owners.

6.2 Proving Statements About Entries in Zero-Knowledge

A *zero-knowledge* proof is an interactive proof in which the verifier learns nothing except the fact that the statement proven is true. Honest-verifier zero-knowledge proofs-of-knowledge protocols exist for proving various statements about discrete logarithms in groups of known order [4,5]. This allows to prove statements about cryptographic primitives that operate in these groups, for instance the knowledge of a commitment or the equality of two commitments' openings. Moreover, note that it is possible to prove AND and OR relations of these statements [6]. Such protocols can be made non-interactive by applying a cryptographic technique called the Fiat-Shamir heuristic [11].

These proof methods are directly applicable to our scheme. Our construction for negative databases based on the Discrete-Logarithm problem allows the parties who know the positive entries, or the data subjects themselves to efficiently prove statements about entries without revealing any information about their positive representations. For example, a user can prove that two entries correspond to his username, and that the sum of two fields is less than a certain threshold. Similarly one can prove that the entry coresponding to their age is larger than a certain minimum age, to gain access some age restricted information.

7 Limitations of Negative Databases

By their very design and properties negative databases have some limitations, and should be used with due care as part of larger privacy enhancing systems. As we have seen, a user can query whether a particular record field value is present or not in the positive database given only the negative representation. Such a

user can only extract additional information by exhaustively enumerating all possible entries.

For many real world databases this may represent a severe weakness. Typical records will include fields that have little variance, such as 'gender' (usually binary) or 'date of birth' (that contains about 7 bits of entropy.) Even fields populated with elements from a theoretically large space, such as names or surnames, may be efficiently enumerated by an adversary that has access to additional information such as population registers or electoral rolls, that are often public. Therefore, by design, a negative database cannot hide such fields.

Many strategies are possible to protect negative databases against such *efficient enumeration attacks*. The first strategy is to systematically aggregate low entropy fields into larger fields. This makes it harder for an adversary to guess them correctly, since the full guess much match, but also does not allow for searches and joins on specific fields.

A second approach would be to include with each low entropy field a high entropy key that is specific to the individual referred to by the field (such as a social security number, or a passport number.) Queries to the database would then need to be appended by the key to be successful, restricting the ability to find records to those that know individuals well enough to have their corresponding key.

8 Conclusion

We have shown practical and efficient schemes to implement negative databases. The security of these schemes is reduced to well understood cryptographic assumptions that have been the subject of considerable scrutiny in the literature. These schemes only occupy $\mathcal{O}(m)$ space and queries are performed in $\mathcal{O}(m)$ time, in comparison with the $\mathcal{O}(m \cdot l)$ space and time complexity for the original proposal. For very large fields our schemes could even achieve a compression of the original database. Records with multiple fields only increase the cost of storage and queries linearly.

The first scheme we show is based on the security of hash functions. Our non-optimised implementation allows for fast queries, with about 2 milliseconds per query for a database of 5000 elements (in worst case queries, i.e. the searched string was not in the NDB.) The query times increase only linearly with the number of entries and we expect that optimised implementation could be used in industrial strength deployed systems to protect privacy. Integrating 'negative' tables in widely deployed Relational Database Managment Systems (RDBMS) would be a significant step forward in deploying privacy enhancing technologies, and our proposal is efficient and economical enough to be the basis for such deployment.

The second construction we propose, based on the DL related assumptions, is less efficient in space and slower than the first. Its advantage is that it can be used by any party knowing the content of some fields to prove a wide range of statements about them in Zero-Knowledge. This allows for building protocols

that offer even higher levels of privacy protection than in the original negative databases proposals. The cost of doing multiple exponentiation is still prohibitive on commodity hardware to allow for wide deployment of this protocols. Yet servers using standard cryptographic hardware could still benefit from its additional properties.

Acknowledgments. This work was partially supported by the IWT SBO ADAPID project (Advanced Applications for e-ID cards in Flanders), the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). George Danezis is funded by a research grant of the Katholieke Universiteit Leuven. Emilia Käsper was partially supported by the FWO-Flanders project nr. G.0317.06 Linear Codes and Cryptography.

References

1. The non-denial of the non-self. The Economist (August 2006)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
3. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast exponentiation with precomputation (extended abstract). In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 200–207. Springer, Heidelberg (1993)
4. Chaum, D., Evertse, J.-H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
5. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
6. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
7. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A strengthened version of RIPEMD. In: Gollmann, D. (ed.) Fast Software Encryption. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
8. Esponda, F., Ackley, E.S., Forrest, S., Helman, P.: Online negative databases. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 175–188. Springer, Heidelberg (2004)
9. Esponda, F., Ackley, E.S., Helman, P., Jia, H., Forrest, S.: Protecting data privacy through hard-to-reverse negative databases. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 72–84. Springer, Heidelberg (2006)
10. Esponda, F., Ackley, E.S., Helman, P., Jia, H., Forrest, S.: Protecting data privacy through hard-to-reverse negative databases. International Journal of Information Security (2007)
11. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)

12. Gong, L., Lomas, T.M.A., Needham, R.M., Saltzer, J.H.: Protecting poorly chosen secrets from guessing attacks. IEEE Journal on Selected Areas in Communications 11(5), 648–656 (1993)
13. Gordon, D.M.: A Survey of Fast Exponentiation Methods. Journal of Algorithms 27(1), 129–146 (1998)
14. International Organization for Standardization. ISO/IEC 10118-3:2004: Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions. International Organization for Standardization, Geneva, Switzerland (February 2004)
15. Lien, R., Grembowski, T., Gaj, K.: A 1 Gbit/s partially unrolled architecture of hash functions SHA-1 and SHA-512. In: Topics in Cryptology-CT-RSA 2004 Proceedings, pp. 324–338 (2004)
16. National Institute of Standards and Technology. FIPS PUB 180-2: Secure Hash Standard. National Institute for Standards and Technology, Gaithersburg, MD, USA, August 2002, Supersedes FIPS PUB 180 1993 May 11 and 180-1 (April 17, 1995)
17. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321 (Informational) (April 1992)
18. Sakiyama, K., Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p). International Journal of Electronics 99(99), 15 (2007)
19. 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)

Author Index