Orr Dunkelman (Ed.)

LNCS 7178

Topics in Cryptology – CT-RSA 2012

The Cryptographers' Track at the RSA Conference 2012
San Francisco, CA, USA, February/March 2012
Proceedings

RSA CONFERENCE 2012

Lecture Notes in Computer Science 7178

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

Orr Dunkelman (Ed.)

Topics in Cryptology – CT-RSA 2012

The Cryptographers' Track at the RSA Conference 2012
San Francisco, CA, USA, February 27 – March 2, 2012
Proceedings

Springer

Volume Editor

Orr Dunkelman
University of Haifa
Computer Science Department
31905 Haifa, Israel
E-mail: orrd@cs.haifa.ac.il

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-27953-9 e-ISBN 978-3-642-27954-6
DOI 10.1007/978-3-642-27954-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012930020

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

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The RSA conference has been a major international event for information security experts since its introduction in 1991, including hundreds of vendors and thousands of attendees with 20 tracks of talks. Among these tracks of the RSA conference, the Cryptographers' Track stands out, offering a glimpse of academic research in the field of cryptography. Founded in 2001, the Cryptographers' Track has established its presence in the cryptographic community as a place where researchers meet with industry.

This year the RSA conference was held in San Francisco, California, from February 27 to March 2, 2012. The CT-RSA conference servers were hosted by the university of Haifa, Israel. This year, 113 submissions were received, a record number of submissions. Out of the 113 submissions, the Committee selected 27 papers for presentation (one of the accepted papers was withdrawn). It is my pleasure to thank all the authors of the submissions for the high-quality research. The review process was thorough (each submission received the attention of at least three reviewers and at least five for submissions involving a Committee member). The record number of submissions, as well as their high quality, made the selection process a challenging task, and I wish to thank all Committee members and the referees for their hard and dedicated work.

Two invited talks were given. The first was given by Ernie Brickell about "The Impact of Cryptography on Platform Security" and the second was given by Dmitry Khovratovich on "Attacks on Advanced Encryption Standard: Results and Perspectives."

The entire Committee, and especially myself, are extremely grateful to Thomas Baignères and Matthieu Finiasz for the iChair software, which facilitated a smooth and easy submission and review process. I would also like to thank Amy Szymanski who worked very hard to properly organize the conference this year.

December 2011 Orr Dunkelman

CT-RSA 2012
The 12th Cryptographers' Track — RSA 2012

San Francisco, California, USA, February 27–March 2, 2012

Program Chair

Orr Dunkelman	University of Haifa, Israel

Steering Committee

Marc Fischlin	Darmstadt University of Technology, Germany
Ari Juels	RSA Laboratories, USA
Aggelos Kiayias	University of Connecticut, USA
Josef Pieprzyk	Macquarie University, Australia
Ron Rivest	MIT, USA
Moti Yung	Google, USA

Program Committee

Adi Akavia	Weizmann Institute of Science, Israel
Giuseppe Ateniese	Sapienza-University of Rome, Italy and Johns Hopkins University, USA
Jean-Philippe Aumasson	Nagravision, Switzerland
Roberto Avanzi	Ruhr-Universität Bochum, and Qualcomm CDMA Technologies GmbH, Germany
Josh Benaloh	Microsoft Research, USA
Alexandra Boldyreva	Georgia Institute of Technology, USA
Carlos Cid	Royal Holloway, University of London, UK
Ed Dawson	Queensland University of Technology, Australia
Alexander W. Dent	Royal Holloway, University of London, UK
Orr Dunkelman (Chair)	University of Haifa, Israel
Marc Fischlin	Darmstadt University of Technology, Germany
Pierre-Alain Fouque	École Normale Supérieure and INRIA, France
Kris Gaj	George Mason University, USA
Marc Joye	Technicolor, France
Jonathan Katz	University of Maryland, USA
Nathan Keller	Weizmann Institute of Science, Israel
John Kelsey	National Institute of Standards and Technology, USA
Aggelos Kiayias	University of Connecticut, USA
Çetin Kaya Koç	Istanbul Şehir University, Turkey and University of California, Santa Barbara, USA

Marco Macchetti
Alexander May
Florian Mendel
Daniele Micciancio
Oliver Mischke
Amir Moradi
Eduardo Morais
Andrew Moss
Debdeep Mukhopadhyay
Tomislav Nad
Samuel Neves
Juan Gonzalez Nieto
Maria Cristina Onete
Elisabeth Oswald
Roger Oyono
Pascal Paillier
Kenny Paterson
Souradyuti Paul
Ray Perlner
Ludovic Perret
Viet Pham
Thomas Plos
David Pointcheval
Elizabeth Quaglia
Christian Rechberger
Alfredo Rial
Thomas Ristenpart
Marcin Rogawski
Werner Schindler
Berry Schoenmakers
Dominique Schröder
Pouyan Sepehrdad
Igor Shparlinski
Rosemberg Silva
Joseph H. Silverman
Martijn Stam
Jaechul Sung
Qiang Tang
Isamu Teranishi
Mehdi Tibouchi
Michael Tunstall
Leif Uhsadel
Jeroen van de Graaf
Rajesh Velegalati
Frederik Vercauteren
Bogdan Warinschi
William Whyte
Kenneth Wong
M.-E. Wu
Keita Xagawa
Panasayya Yalla
Ching-Hua Yu

Table of Contents

Side Channel Attacks I

Digital Signatures I

Public-Key Encryption I

Cryptographic Protocols I

Secure Implementation Methods

Symmetric Key Primitives

Side Channel Attacks II

Digital Signatures II

Cryptographic Protocols II

Public-Key Encryption II

Side Channel Attacks III

Secure Multiparty Computation

Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures

An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism

Amir Moradi, Markus Kasper, and Christof Paar

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{moradi,mkasper,cpaar}@crypto.rub.de

Abstract. This paper presents a side-channel analysis of the bitstream encryption mechanism provided by Xilinx Virtex FPGAs. This work covers our results analyzing the Virtex-4 and Virtex-5 family showing that the encryption mechanism can be completely broken with moderate effort. The presented results provide an overview of a practical real-world analysis and should help practitioners to judge the necessity to implement side-channel countermeasures. We demonstrate sophisticated attacks on off-the-shelf FPGAs that go far beyond schoolbook attacks on 8-bit AES S-boxes. We were able to perform the key extraction by using only the measurements of a single power-up. Access to the key enables cloning and manipulating a design, which has been encrypted to protect the intellectual property and to prevent fraud. As a consequence, the target product faces serious threats like IP theft and more advanced attacks such as reverse engineering or the introduction of hardware Trojans. To the best of our knowledge, this is the first successful attack against the bitstream encryption of Xilinx Virtex-4 and Virtex-5 reported in open literature.

1 Introduction

The market for digital electronics is highly competitive. Thus, a new product has to provide a unique selling point to be successful. It can be technologically superior to other products, or convince with a better design, better quality, or appealing price. But having invested a lot of efforts and money into the development of a new product, which options does a company have to prevent competitors from stealing or even cloning the product? On the legal side, patents offer a handle to stem against many kinds of product piracy threats. However, patent registration and monitoring is an expensive task that comes with many pitfalls. Furthermore, patents expose know-how to competitors, and patent disputes often imply a high financial risk. Besides that, not all technology qualifies for patent registration. In this case a manufacturer only receives legal protection within the scope of copyrights. At the end of the day the throughout protection of intellectual property (IP) remains a task that needs to be considered at a technological level.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 1–18, 2012.

In our work we analyze the provided IP protection mechanism of Xilinx's Virtex-4 and Virtex-5 Field Programmable Gate Array (FPGA) families called bitstream encryption. We provide a detailed description of this real-world side-channel analysis, illustrating the steps required to perform a black-box analysis of a mostly undocumented target, i.e., the embedded decryption module. Our results provide many practical insights that are of avail for practitioners in industry and academia. They will learn to judge the feasibility of possible side-channel analyses and to evaluate the black-box side-channel security of electronic devices in a realistic attack scenario.

1.1 Content of This Paper

The paper is organized as follows. Next, we give a short overview of FPGAs and their security features. Here we mainly focus on Xilinx's bitstream encryption solution employed in the analyzed Virtex FPGAs and the consequences of a successful bitstream extraction. In Section 4 we provide an introduction to the employed attack method and detail the practical issues we encountered to mount our attack. Then, in Section 5 we have a short glance on the computing architecture followed by our experimental results and the final conclusion.

2 Introduction to FPGAs

Designers of digital embedded systems have three different technology options to choose from. These are application specific integrated circuits (ASICs), microcontrollers and FPGAs. ASICs are specially-tailored pieces of silicon that realize exactly the desired functionality. They provide the highest performance and are very cost efficient when produced in large quantities. On the other hand, ASICs implement a static functionality that can not be modified or updated once produced. Microcontrollers are a class of silicon devices that implement a fixed instruction set allowing the designer to write pure software programs. This offers a good flexibility and allows for updates of devices in the field. On the other hand, a microcontroller's performance is limited by its nature of sequential instruction processing. The third option available to designers, the FPGAs, close the gap between powerful but inflexible ASICs and highly flexible but performance-limited microcontroller-based solutions. An FPGA is an integrated circuit that consists of many configurable logic blocks (CLBs), which can be configured to represent basic digital design elements as, e.g., logic gates and registers. In addition, the connections inside an FPGA are configurable such that inputs and outputs of several building blocks can be connected to each other.

Similar to an ASIC, a designer of a digital system provides a hardware description language (HDL) representation of the digital design that the FPGA should implement. Modern FPGAs come with additional built-in functionality, such as RAMs, adders and multipliers or even complete microcontroller cores that can be included into the design. Instead of mapping the design to transistors in silicon to manufacture ASICs, the development tools for FPGAs will map the HDL representation to CLBs and routed connections. Developing for an

FPGA is in fact very similar to the development of a microcontroller software. The difference is that the software will be executed by the microcontroller in a sequential fashion while the configuration of an FPGA allows for highly parallel designs. The software equivalent for FPGAs is called *bitstream* or *configuration*. There are multiple ways to store the configuration for an FPGA. Since their first generation, FPGAs use SRAM (Static Random Access Memory) to store the configuration. This requires that the FPGA has to be reconfigured after each power loss. In this setup, the configuration data is stored in an external non-volatile ROM (Read Only Memory) and is loaded on each power-up. Although today there are also devices that provide an internal Flash, EPROM, EEPROM or even one-time programmable memory (fuses), the market is still dominated by FPGAs using an external configuration ROM.

Being in general re-programmable, FPGAs offer the same flexibility as microcontrollers. On the other hand, they can achieve a much higher performance enabling many new applications that otherwise would require the power of an ASIC. FPGAs allow a fast time to market and many design iterations within a few hours. Therefore, FPGAs are often also used to test the digital functionality of ASIC designs in form of FPGA prototypes. Overall, FPGAs are more expensive than ASICs in medium to high volumes.

3 FPGA Security

Coming back to the initial question of IP protection we now introduce the general vulnerabilities arising from the existence of machine readable configuration files of FPGA designs.

3.1 Bitstream Vulnerabilities

During power-up, an SRAM-based FPGA reads its configuration from an external non-volatile memory. The configuration includes the functional design as well as the I/O configuration for the pins and the exact placement and routing of all used components. The whole design of an FPGA application is encoded within the configuration file, the role of which can be considered similar to the role of software for microcontrollers. As introduced before, this nature provides a means to update the configuration file of an FPGA to adapt its behavior to new system requirements or to fix early design flaws.

On the other hand, it also gives rise to the fact that the bitstream needs to be considered as the key element of an FPGA design and thus requires protection. In this section we discuss the impact of attacks on an unprotected bitstream and provide a glance at the broad scope of possible consequences.

Consider a company that just released a new product. An adversary will have easy and anonymous access to hardware, once the product is released. Thus, it seems reasonable to assume that a determined adversary will be able to find a way to access the plain configuration file, unless it has been protected by means of IP protection mechanisms. For now we ignore any possible protection and restrict our discussion to unprotected bitstreams. The bitstream extraction

could be performed by eavesdropping the configuration process, unsoldering the configuration ROM or downloading a firmware update that includes a new FPGA bitstream. An adversary who stole a bitstream file can copy it and thus steal IP (Intellectual Property). This opens doors for product cloning and product piracy. The pirated products could be brought to the black-markets to earn money, without the need for the adversary to invest in product development. This causes damage in several ways. The IP-owning company would suffer from the lost sales directly due to the loss of income. But furthermore, the pirated products may be of weak quality and thus pose the additional threat to cause image loss. Even more, the cloned products might also come with additional functionality (e.g., offering a remote control) or a fancy optical design, such that there could even be seriously competing products originating from pirated hardware. In this case it might be hard to prove or even find that IP has been stolen and cloned.

Cloning FPGAs has even more serious implications in security sensitive scenarios, e.g., military technology like nuclear warheads or surveillance satellites. Adversaries that are able to extract the FPGA configuration would get access to highly sensitive technology. The possibility of bitstream reversal and manipulation that are discussed next, makes these scenarios even worse.

For many years people challenged the bitstream's security with respect to bitstream reversal. The motivations to do this are manifold. Some want to develop their customized toolchain for FPGAs and thus need to be able to compile an HDL design to a valid configuration file. Others are security researchers that search to extract secret algorithms or keys from the bitstream, and other parties might be interested in stealing a competitors technology. A good judgment of the difficulty to reverse engineer a bitstream is also essential for security evaluators and system designers that aim at increasing product security by selecting a suitable combination of secure devices and anti-tamper technologies to minimize the risk of vulnerabilities. Bitstream reversal can be used for proving infringement as well. These examples illustrate that both criminals and designers have a decent motivation to reverse engineer bitstreams.

The most studied bitstream format is that of the FPGA world market leader Xilinx, Inc. The Ph.D. thesis of Saar Drimer provides a good overview on FPGA security [4]. For lack of space, we only sum up some small parts of his discussion on bitstream reversal, and refer the interested reader to his work for all details. The general description of the structure of the bitstream can be found in Xilinx publicly available documents [18,20,21,22]. Ziener *et al.* have shown [23] that the configuration of look-up tables (LUT) and RAM contents can be extracted from bitstreams with moderate efforts. According to Drimer, methods translating a bitstream to a netlist are not technically mature, yet. In open literature there are two works documenting notably successful reversals of bitstreams. The first is the free software project "Ulogic" by Note and Rannaud described in a report by the developers [12]. This work relates Xilinx Design Language (XDL) plaintext representations of placelists to bitstream bits, with a result that, as stated by the authors, is still a step away from a true netlist. The related "FPGA analysis tool"

by Kepa *et al.* [7] adds a graphical representation to the decoded bitstreams and provides another step towards true reverse engineering. As the encoding of bitstreams is undocumented but not confidential by means of cryptography, it is a strong belief in industry and academia that bitstream reversal of FPGAs may be a difficult and time consuming, but nevertheless a technically-feasible task.

The possibility to reverse engineer a bitstream lets arise even further threats. Besides cloning and stealing IP, the reverse engineering of a bitstream also allows modifying the designs. This way Trojan Hardware could be added to a security-sensitive system. This malicious circuitry may, e.g., implement a hidden backdoor or some kind of kill-switch functionality to an FPGA that implements a sensitive application (e.g., nuclear power plant, military- or satellite technology or a banking application). Besides this, the option to modify a design also allows hobbyists to customize commercially available hardware to add functionality or improve performance. Beyond the discussion of the bitstream security, the interested reader is referred to [4] for a throughout evaluation of many aspects of FPGA security and to [2] where invasive attacks on FPGAs are discussed.

3.2 IP Protection for FPGAs[1]

As of 2001 [16], Xilinx implemented an encryption mechanism in many of its recent FPGA series released within the last decade to counter these threats. This mechanism is called *bitstream encryption* and works in the following way: instead of storing a plain bitstream file within the configuration ROM, the designer encrypts the bitstream configuration beforehand. The encryption – using AES-256 in CBC (Cipher Block Chaining) mode for the Virtex-4 and Virtex-5 FPGAs – is performed in software by the Xilinx ISE development tools. The used key is chosen by the designing engineer and is programmed into the Virtex FPGA. The part of the FGPA memory storing this secret key is battery-powered so that the key will immediately be erased on power loss of the battery support. This feature is designed to hinder invasive attacks to recover or reverse engineer a device configuration.

With the known encryption key inside the FPGA and the encrypted bitstream stored within a ROM, products can securely configure the FPGAs as only AES-256 encrypted data passes the channel between ROM and FPGA. The FPGA has a dedicated AES hardware to decrypt the bitstream. This hardware is not accessible for other purposes within the FPGA due to export regulations of cryptography.

3.3 Real-World Attacks

With this mostly theoretical discussion on several threats and countermeasures for FPGA applications, the remaining question is whether the manufacturer's countermeasures are able to provide the advertised protection in the real world,

[1] Due to page restriction we limit this discussion to only the scheme provided by Xilinx FPGAs.

i.e., successfully prevent attacks. Unfortunately decisions on implemented countermeasures are in most cases driven by economical reasons. Thus, products in industry will only be guarded against risks and threats when customers are expected to be willing to pay for the additional security. Thus, often well-known security risks from academic literature are not considered when designing commercial products, as long as there is no evidence for real-world implications. One class of these attacks often underestimated in industry are the side-channel attacks introduced in the next section. In our contribution we analyze the IP protection mechanisms of recent FPGAs with respect to side-channel security. We show that the implemented features of the studied products can be broken with moderate efforts and thus fail to protect the implemented configuration file.

4 Side-Channel Analysis Attacks

4.1 Introduction to Side-Channel Analysis Attacks

Today Side-Channel Analysis (SCA) is a mature field in applied security research. Differential side-channel analysis methods have been introduced first by Kocher et al. around 10 years ago [8]. Since then the field has grown rapidly and many new tools and distinguishers for side-channel analysis have been evaluated. In reply to the new threat developed in the scientific literature many countermeasures have been proposed, implemented and broken. Also, experts from the field of theoretical cryptography recognized side-channel attacks as an important topic seeding a community of researchers working on general leakage resilience and provable security bounds for side-channel countermeasures. Beyond purely academic studies, side-channel attacks and reverse engineering have been shown to also have real-world impact. Examples are the attacks on NXP's Mifare Classic devices [11], a bouquet of attacks on Microchip's KeeLoq remote keyless entry systems (primary article [5]), and recently also SCA attacks on Mifare DESFire contactless smartcards [15]. Lately a successful side-channel key recovery attack on the bitstream encryption feature of the older Xilinx Virtex-II pro FPGAs, which employ 3DES as the decryption engine, has been reported in [10]. In this paper we describe a practical side-channel analysis attack on the bitstream decryption engines of the more recent Virtex-4 and Virtex-5 FPGAs. These attacks demonstrate that industrial products in fact require to implement side-channel countermeasures and that side-channel attacks are not a pure academic playground but have a real-world impact on the security of embedded systems.

The method employed in this work is a sophisticated type of Correlation Power Analysis (CPA) as first introduced in [3]. In this method the power consumption or electro-magnetic radiation (EM) of a device is measured while executing a cryptographic algorithm. In addition to the physical power consumption of the analyzed device, also the communication of the device is eavesdropped to get access to the ciphertexts (or plaintexts) that will be (or have been) processed. In our case the ciphertexts, i.e., blocks of the encrypted bitstream, are available by eavesdropping the configuration process and the analyzed cryptographic primitive is an AES-256 decryption module.

During the analysis itself the known ciphertexts are used to predict an intermediate value processed by the AES algorithm. A hypothetical intermediate value for each trace is calculated assuming a fixed subkey[2]. In the next step these hypothetical values are used in a hypothesis test, which allows distinguishing the key used by the device from wrong key hypotheses. In a CPA attack the used distinguisher is Pearson's correlation coefficient estimated by the sample correlation.

Side-channel analysis attacks follow a divide-and-conquer strategy. That is, the key is recovered in small pieces. Typical attacks use subkeys of 8 (AES) or 6 (DES) bits and target S-box outputs.

In our attack we can use a full bitstream as a set of multiple ciphertexts. In order to apply the correlation distinguisher, the predicted intermediate values have to be mapped to hypothetical power consumptions, which will then be compared with the measured power consumption. For hardware designs a reasonable choice to do so is the Hamming distance (HD) model, which counts the number of bits of an intermediate value that are toggled within a clock cycle.

4.2 Measurement Setup

We have started our analysis by examining a Virtex-4 FPGA. We have used a "Virtex-4 FF668 Evaluation Board" [19], which provides a ZIF socket to host Virtex-4 devices with FF668 packaging. Since the board has not been designed for side-channel analysis, we have placed a resistor in the V_{CCINT} path and removed the blocking capacitors[3]. There are three different V_{CC} paths in Virtex-4 FPGAs: V_{CCINT} (1.2V) as the power pin for internal core circuits, V_{CCAUX} (2.5V) as the power pin for auxiliary modules, and V_{CCO} (1.2∼3.3V) as the power pins for the output pin drivers. We analyzed all power pins, but similar to the results reported in [10], the successful results were obtained when considering the power traces measured in the V_{CCINT} path.

Our target Virtex-4 FPGA model was an XC4VLX25, and the power traces were captured using a LeCroy WP715Zi digital oscilloscope at a sampling rate of 2.5 GS/s and a LeCroy AP033 active differential probe. We have also designed a microcontroller based module which configures the FPGA in slave serial mode (see [20] for more details on Virtex-4 configuration modes). It communicates with a PC and passes the bitstream chunks[4] to the FPGA. The same board also provides a trigger pin to start the oscilloscope each time right before sending a bitstream chunk to the FPGA. The acquisition of power traces started after sending the header part of the bitstream. Each measured trace belongs to the previously sent bitstream chunk.

[2] By subkey we denote the part of a key that has an effect on the predicted intermediate value.

[3] This task is essential and common when performing power analysis attacks on real-world devices, as the capacitors would filter the analyzed signal.

[4] Since the Virtex-4 bitstream encryption uses AES-256, we define each block of 128 bits as a chunk.

4.3 Introductory Experiments

In a real world attack, it is of major importance to work very accurately and carefully to make sure the chosen method and all employed models are suited for the analysis. Thus, some preliminary work is required to eliminate uncertainties wherever possible. The first step in side-channel analysis is to find the correct instance in time when the targeted security primitive (here the AES-256 decryption) is processed. We created a very simple design using the Xilinx ISE development tools and generated both, the corresponding bitstream and its encrypted counterpart. The 128-bit IV (Initialization Vector)[5] and the 256-bit key used to generate the later one were loaded into the FPGA[6], whose V_{CCBAT} pin was continuously battery powered at 3.0V. Using the public documentation by Xilinx ([18,21]) we verified the order of the bits and bytes of the ciphertexts within the encrypted bitstream.

Comparing the power traces corresponding to the plain and the encrypted bitstreams let us to identify the interesting time instances. Two exemplary power traces are shown in Fig. 1(b) and Fig. 1(c). Due to the high level of noise present in the measurements we employed mean traces instead of raw measurement data in this step, i.e., each mean trace has been obtained by calculating the average of 10 000 traces. The mean traces, plotted in Fig. 1(d) and Fig. 1(e), show clear differences between the configuration with an active decryption module and the unencrypted configuration. We identified 26 clock cycles that show significant differences.

We should emphasize that in contrary to the Virtex-II case [10] (in which the full 3DES decryption is executed after a certain positive edge of the configuration clock signal[7]), the computations of the AES decryption rounds are spread and activated by consecutive positive edges of the corresponding configuration clock signal. Thus, in the case of the Virtex-4 the performance of the decryption module has much less impact on the maximum frequency of the configuration clock signal, as just single rounds need to be processed within a configuration clock cycle. According to the public documentation [17], when a Virtex-II is configured in SelectMAP mode using an encrypted bitstream, the BUSY signal has to be monitored[8] to ensure that the decryption module is ready for next data. The Virtex-4 FPGAs do not need to drive the BUSY signal during configuration, even when configuring using the maximum frequency and an encrypted bitstream [20]. In summary, our first experiment allowed us to find the most valuable instances in time for our SCA and gave us a hint towards a most likely round-based architecture.

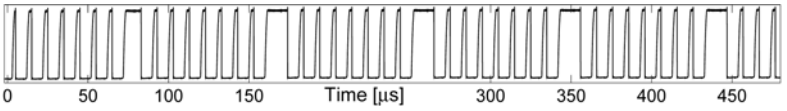
A close look at a power trace (Fig. 1(f)) reveals that the measurements include HF-modulated waveforms. Therefore, we used a Chebyshev low-pass filter to

[5] The IV is used as initial value in CBC mode.

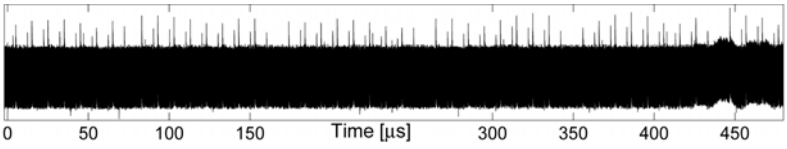
[6] Note that the only way to load the encryption key is through the JTAG interface [20] and by means of a standard configuration device.

[7] TCK in the case of JTAG and CCLK in the other configuration modes.

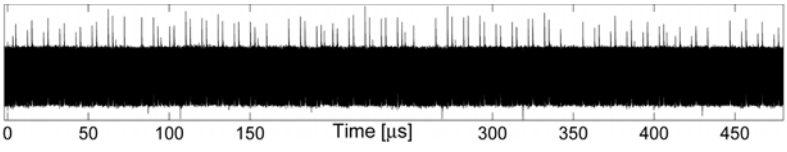
[8] The SelectMAP mode enables sending 8 bits of the bitstream at each clock cycle, and BUSY is one of relevant handshaking signals.



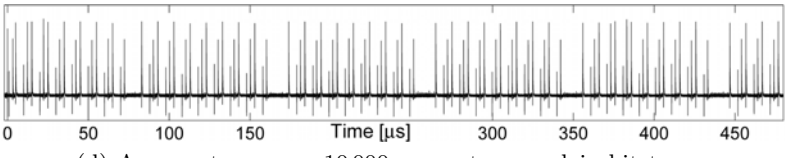
(a) Configuration clock signal



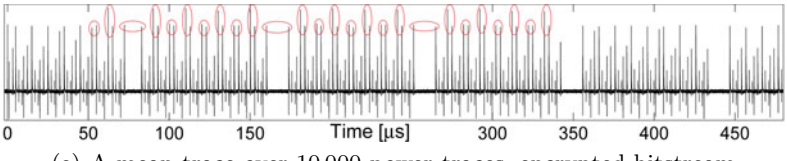
(b) A sample power trace, plain bitstream



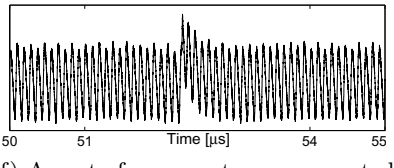
(c) A sample power trace, encrypted bitstream



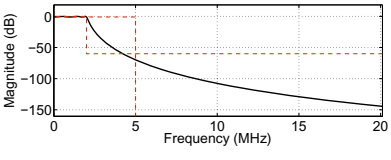
(d) A mean trace over 10 000 power traces, plain bitstream



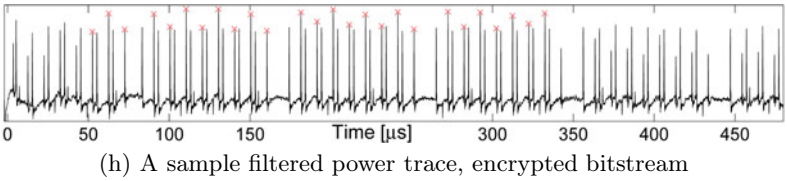
(e) A mean trace over 10 000 power traces, encrypted bitstream



(f) A part of a power trace, encrypted



(g) Magnitude Response of the used filter



(h) A sample filtered power trace, encrypted bitstream

Fig. 1. Virtex-4: Sample power traces, mean traces, and result of filtering

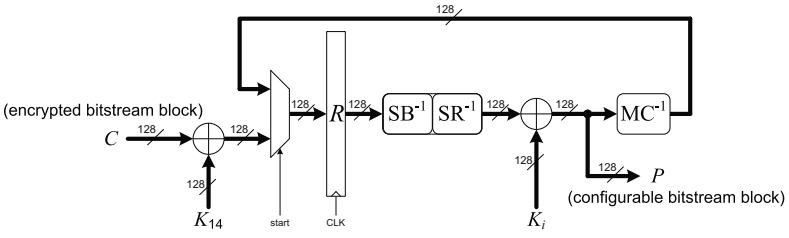


Fig. 2. The employed model of the internal architecture of the AES-256 decryption module

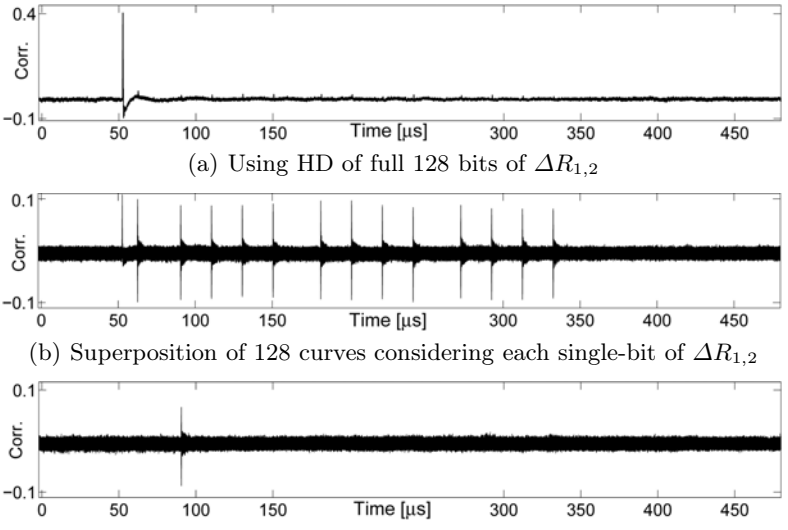
reduce the effect of the high frequency components. The configuration of the used filter and the result of the filtering of a sample trace are shown in Fig. 1(g) and Fig. 1(h) respectively.

With this initial analysis and preprocessing of our measurements a remaining task was to find and verify a model for the internal architecture of the AES-256 decryption module, that allows us to relate the measured power consumption to the processing of the decryption primitive. The method that we used is to correlate the filtered power traces to predictions based on a hypothetical power model of the architecture. Thus, by trial and error, we guessed several possible architectures and modeled their power consumption. Using the known key we applied the models to the encrypted bitstream and correlated the resulting hypothetical power values to our measurements. In this experiment a significant correlation indicates a valid power model that might be a candidate to be used in the following cryptanalysis. Having examined several architectures and a couple of hypothetical power models, the only working model we found was the combination of the architecture shown by Fig. 2 and a HD model targeting the 128-bit register R.

Figure 3 shows the results of correlating the bit flips of the intermediate register between the first and second decryption rounds. More precisely, Fig. 3(a) has been obtained computing Pearson's correlation coefficient between each time instance of the filtered power traces and HD of register R in the first and second decryption rounds, i.e., Hamming weight (HW) of

$$\Delta R_{1,2} = \Big[\underbrace{C \oplus K_{14}}_{R_1}\Big] \oplus \Big[\underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_1)\big) \oplus K_{13}\Big)}_{R_2}\Big],$$

where C, K_{14} and K_{13} represent ciphertext, round key 14 and round key 13 respectively. Also, MC^{-1}, SB^{-1} and SR^{-1} are abbreviations for InvMixColumns, InvSubBytes and InvShiftRows transformations. The high peak at around 52μs indicates a very high dependency between the measured power traces and the intermediate values in our considered internal architecture. This time instance corresponds to the positive edge of the configuration clock signal at the sixth clock cycle (see Fig. 1(a)). In order to find which bit flips in register R causes the most significant correlation, we repeated the same computation considering each



(a) Using HD of full 128 bits of $\Delta R_{1,2}$

(b) Superposition of 128 curves considering each single-bit of $\Delta R_{1,2}$

(c) Superposition of 128 curves considering each single-bit of $\Delta R_{2,3}$

Fig. 3. Virtex-4: Results of correlating the filtered power traces to the bit flips in the intermediate register R

bit of $\Delta R_{1,2}$ independently. This led to the 128 curves shown in Fig. 3(b). For some yet unknown reasons, additional high peaks also appear in 13 other time instances. The results of applying a similar procedure in the next decryption round, i.e., using

$$\Delta R_{2,3} = R_2 \oplus \Big[\underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_2)\big) \oplus K_{12}\Big)}_{R_3}\Big],$$

are depicted in Fig. 3(c). Notably here the high peaks only appear at one single time instance, i.e., 90μs corresponding to the start of the 9th configuration clock cycle. The curves in Fig. 3 have been derived using the power traces of the 60 000 decryptions performed during a single power-up of the FPGA. We should emphasize that no significant peak appeared when we continued this procedure for the next decryption rounds. In fact, it seems that our guess about the architecture does not completely match with the target internals. Nevertheless, according to the results these assumptions are adequate to successfully perform the attacks as shown later.

4.4 Implemented Attack

Using the extracted information about the leaking points and the architecture the straightforward way to perform an attack is to guess parts of two consecutive round keys K_{14} and K_{13}, and then use the single-bit power model to predict

the key dependent leakage of the subsequent rounds. Due to the structure of InvMixColumns, at least one column (32 bits) of each round key has to be guessed at each step of the attack, which means searching the large key space of 2^{64}. However, because of the linear property of InvMixColumns one can write R_2 as

$$\underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_1)\big)\Big)}_{R'_2} \oplus \underbrace{\mathrm{MC}^{-1}\Big(K_{13}\Big)}_{K'_{13}}.$$

Moreover, since K_{13} and consequently K'_{13} are fixed and independent of the ciphertexts and decryption intermediate values, they can only change the polarity of our considered single-bit power model, i.e., bit flips of $\Delta R_{1,2}$. Therefore, guessing a column of K_{14}, i.e., searching a space of 2^{32}, is adequate when the single-bit power model is used. Note that in this case, one cannot take a power model using more bit flips, e.g., HD of whole 32 bits, in a CPA attack. However, a Mutual Information Analysis [6] or a multi-bit DPA [9] may be feasible.

We should highlight that one can decrease the search space of the attacks to the space of 2^8 by a chosen ciphertext scenario. For this, parts of R'_2 will be fixed as long as the corresponding ciphertext bytes are fixed. However, configuring the FPGA using a wrong encrypted bitstream (caused by the chosen ciphertexts) results in forbidden interconnections of internal wires, e.g., connecting two output pins to each other. Thus, each configured invalid bitstream block causes additional leakage currents, which lead to additional interferences with the measured side-channel signal. More importantly, these currents also heat up the FPGA and may even damage it. For this reason we did not pursue this approach and stuck to the approach using a valid encrypted bitstream. Nevertheless, when following the chosen ciphertext approach, the destructive effect of invalid bitstream chunks can be limited by resetting the configuration process after measuring a certain number of power traces, e.g., after each eighth chunk.

As a result, a full 128-bit K_{14} can be recovered by performing four attacks, each of which independently recovers a 32-bit part of the key. Note that in order to recover the full 256-bit key of AES-256, one needs to extract two consecutive 128-bit round keys, e.g., here K_{14} and K_{13}. We therefore need to extend the attack on the next decryption round. R'_2 can be computed for every ciphertext knowing K_{14}, and one can write

$$\Delta R_{2,3} = R'_2 \oplus K'_{13} \oplus \underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_2)\big)\Big)}_{R'_3} \oplus \underbrace{\mathrm{MC}^{-1}\Big(K_{12}\Big)}_{K'_{12}}.$$

As before, linear contributions of key bits, i.e., K'_{12} and K'_{13} can be omitted in our single-bit power model (here single bit flips of $\Delta R_{2,3}$). The attack described above can be run to recover the round key K_{13} which influences the hypotheses due to its contribution to R_2. Note that each part of this attack again recovers only a 32-bit column of K'_{13}. Knowing all bits of K'_{13} allows computing K_{13} by applying the MixColumns transformation. The result of the key extracting attacks and more details about their efficiency are given in Section 5.2.

4.5 Countermeasures

Today there exists a set of countermeasures that is believed to provide enough protection against side-channel attacks, that they can be considered secure for most practical purposes. More precisely, the reached level of security is boosted to a certain level which makes practical attacks not impossible, but infeasible in practice. Unfortunately most of these methods are patent-protected and thus often avoided in industry due to the involved royalties. Furthermore, many people in industry still recognize side-channel attacks as academic playground without any real-world impact and thus do not see the necessity of side-channel countermeasures for their products.

5 Implementing the Attack

5.1 Employing nVidia's CUDA

Our attack needs to perform an overall of eight analyses each statistically evaluating a set of 2^{32} key candidates. For each key candidate a hypothetical intermediate needs to be calculated for each used power trace. Fortunately, the locations of the occurring leakage we found earlier allowed us to limit the attack to a single time instance per decryption round. To cope with the large amounts of computations we employed NVidia's CUDA architecture[9] [14], to speed up our attack using the parallel computing capabilities of modern GPUs (Graphic Processing Unit). The used server was equipped with four NVidia Tesla C2070 cards [13], each having around 6 GB of memory and 448 thread processors arranged as 14 streaming multiprocessors. The implemented kernel processed one key per thread and was launched using a granularity of 256 threads per block and a $(64, 256, 256)$ grid. The resulting 2^{30} 32-bit floating point correlation coefficients per card were then stored to the machine's hard drive for visualisation in, e.g., MATLAB. Using CUDA allowed us to perform our analysis on a single point of 60 000 filtered power traces at a speed of one column each 33 minutes, i.e., an overall runtime of the computations of 264 minutes for attacking all 8 columns of the first two round keys K_{14} and K'_{13}. The corresponding correlation coefficients for the full attack require 128 GiB of hard disk space.

5.2 Attack Results

Amongst the available 32 bits of the register R that are suited to attack a column of the analyzed round key, we have selected the one which shows the highest absolute correlation at the datapoint at $52\mu s$ (see Fig. 3(b)). More precisely, the seventh LSB of the second byte of each column was selected in our attacks on the first decryption round. The result of the attack on the first column of the first round is shown in Fig. 4. Using 60 000 measurements, the highest correlation of

[9] In the following we employ NVidias terminology of threads, blocks and grids as introduced in [14].

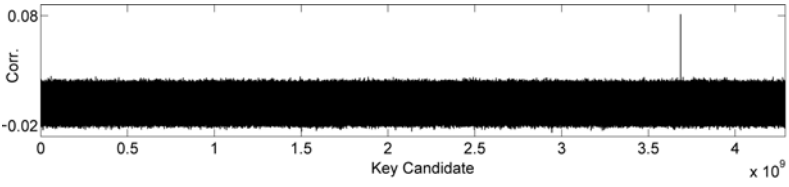


Fig. 4. Virtex-4: Result of an attack on the first column of the first decryption round using bit flips of the 7th LSB of the second byte

0.081 for the correct key candidate can already be clearly distinguished from the wrong key candidates. The next highest correlation value is already as low as 0.025. The attack on the second decryption round was performed in exactly the same way; even the same target bit was selected for the power model. As the results of the analysis of the second round closely reflect the results provided for the first decryption round, we refrain from providing extra figures at this point.

As mentioned before, we have used the measurements corresponding to only one power-up of the FPGA. The amount of possible measurements of each power-up depends on the size of the FPGA fabric (not to the used-defined design). The "Configuration Array Size" of the FPGA [20] defines how many 32-bit words have to be configured by a bitstream. Since 243 048 configurable words are available in our target, 60 762 traces can be measured using a single power-up. From the smallest Virtex-4 FPGA, XC4VLX15, 36 900 traces can be acquired during one power-up. Note that in the case of a high noise level the measurement process can be repeated with the same encrypted bitstream, i.e., with the same ciphertext values, and therefore provide more traces if required.

5.3 Differences to Virtex-5

We have examined the decryption module of a Virtex-5 FPGA re-employing the introduced analysis developed for the Virtex-4 device. The targeted FPGA model was an XC5VLX50 embedded on a SASEBO-GII [1]. We were able to reuse the measurement setup introduced before with minor modifications: Since the serial configuration pins used for the Virtex-4 were not present on the SASEBO-GII board, our microcontroller module was adapted to support configuration via JTAG interface, which is the only available configuration port on the used platform.

Compared to Virtex-4, the main difference was that the attack on the Virtex-5 FPGA required more power traces to be successful, which is mostly due to a worse signal-to-noise ratio due to a newer process technology (i.e., 65 nm instead of 90 nm). In our attacks we have used 90 000 traces[10] acquired during a single power-up of the FPGA, but using more traces of multiple power-ups can still improve discriminability of the correct key hypothesis. To deal with

[10] Our Virtex-5 target FPGA allows for measuring 98 031 traces during a single power-up.

the worse measurement conditions, we have acquired power traces with a sampling rate of 20 GS/s and low-pass filtered the data as before. Approximately the same results as Fig. 1 were obtained, i.e., comparing the mean traces of the plain and encrypted bitstreams showed differences in the same positive edges of the configuration clock signal. Also, correlating the filtered power traces with the single-bit power model considering the same internal architecture led to the similar results shown in Fig. 3 but with lower correlation value, i.e., 0.05 and 0.03 for the first and second decryption rounds respectively. The analysis runtime increased due to the higher number of power traces to around 49 minutes per column or around 6.5 hours for the overall computing time. We should emphasize that analyzing the pure Virtex-5 without having the knowledge obtained during the analysis of Virtex-4, would have been a much more challenging task with more uncertainties.

6 Conclusion

Today, industrial spying and technology theft are a major threat for both companies and government-run facilities. Companies are mostly concerned about IP theft and product piracy and the inflicted losses. Government institutions, on the other hand, need to protect military secrets as well. Our attacks show that the IP protection mechanism of the FPGA world market Xilinx, Inc. can be circumvented using moderate efforts.

Our presented approach allows us to read out the configuration data of Virtex-4 and Virtex-5 devices in the field, leading to the consequences elaborately discussed in Section 3.1. Manufacturers of high-security products and security evaluation labs are well aware of the side-channel vulnerabilities. Therefore, they ensure that additional security countermeasures to protect devices are implemented where necessary. Techniques include for example shielding and molding the electronic circuit to provide additional tamper resistance and therefore deny power or EM measurements. Unfortunately this awareness does not cover all manufacturers of security sensitive devices yet.

We want to underline that this attack targets Xilinx's bitstream encryption engine, and not a third party crypto-implementation inside an FPGA. Although reading out and interpreting the bitstream might also annihilate the security targets of an FPGA design, there is an important difference between a vulnerability in the bitstream encryption and a vulnerability in an implemented primitive. An engineer developing an FPGA design has no influence on the security of the bitstream encryption and thus also no option to improve it. In other words, up to now it is the FPGA manufacturer's responsibility to provide secure IP protection methods. This is slightly different to the microcontroller scenario. Designers using microcontrollers often have the freedom to implement customized bootloaders, that might, e.g., add encryption functionality to the programming. Nevertheless, the microcontroller's manufacturer also has to ensure that the memory including the bootloader and all its secrets cannot be read out.

There are many new insights from this attack. This is the first case to our knowledge, where it was possible to probe and compare the security of subsequent

technology generations of an embedded system in a real-world environment. In this attack we were able to practically verify that an attack on more recent technology nodes still scales within feasible bounds. Furthermore, we were able to show that developing an attack tailored to one product can threaten the security of another product, when a security design is being reused. In our case the analysis of the Virtex-4 allowed us to study the architecture with much less efforts than having to perform the same analysis on the Virtex-5 device. In consequence, we suggest to limit the reuse of security designs, such that the security of a newer product is not lowered by an easier attack on an older product.

Another argument we practically disproved is that attacks on intermediate values that require large key hypotheses are infeasible in practice. We have shown that with todays available computing power an analysis on 60 000 power traces using 32-bit key hypotheses can be performed in less than 4.5 hours. We also explained the different steps that needed to be done to execute a black-box analysis. These differ from purely academic studies, as they include many additional steps as identifying and filtering unknown additional noise sources, the identification of the time instances that need to be considered in the attack and the deduction of a valid model of the implemented architecture. To our knowledge there is no published real-world side-channel attack with a similar attack complexity. Therefore, this work provides an update on the lower bound of attacks that should be considered a realistic threat to real-world systems.

The presented approach practically illustrates that side-channel attacks on real-world systems do not require any detailed knowledge of the implemented architecture. Thus, the extend to which confidential details on the implemented architecture can raise the difficulty of black-box side-channel analyses should not be overestimated. Nevertheless, it remains an open research problem to evaluate the security gain achieved by applying additional confidential obscurity measures as transformations of the plaintexts prior to encryption.

Finally, the most exciting question is to ask why this attack was possible at all. Side-channel analyses are known for more than ten years, and the same holds for bitstream encryption protection mechanisms. Why did Xilinx not implement the available countermeasures? As stated before, it is most likely due to an economic reason. In this case the FPGA configuration has been protected by means of an encryption mechanism. Customers accepted the solution without having the expertise to recognize the obvious possibility of side-channel attacks, and thus did not give rise to a market demand for a side-channel resistant configuration solution. The fact that Xilinx's bitstream encryption has not been broken in public literature for around a decade shows that side-channel attacks on real-world targets, i.e., black-box attacks, just became mature within the last years. From our point of view a prominent problem in security technology is that both, customers and manufacturers, are not aware of the security risks that come with unprotected implementations of cryptographic primitives in embedded systems. On the other hand, those that are aware of the existence of side-channel attacks, often consider them as a purely academic threat without any real-world counterpart. We see an urgent need to change this wrong perception and to recognize the rapid advances

in the minatory field of black-box side-channel analysis. A general guideline should be that cryptographic routines in embedded systems without SCA countermeasures should be considered insecure for all applications where a successful attack can give rise to financial benefits.

To assist hardware manufacturers, scientific research should in the future aim at developing mechanisms beyond mere SCA resistance to face the increasing threat of physical attacks. This could be, e.g., protocols and measures that limit the effect of successful side-channel attacks. In the case of the bitstream encryption a solution avoiding repetitive use of the same key in CBC mode, e.g., by means of some obscure key transformation, would have significantly hardened the analysis. In this context researchers should also reconsider to add obscurity measures in combination with the well-proven crypto primitives to raise the SCA protection of their systems. This would require an attacker to first overcome an additional reverse-engineering step, before being able to analyze the system.

Acknowledgments. The authors want to thank Alessandro Barenghi from Politecnico di Milano for providing the programming tools needed to transfer our attacks to the Virtex-5 FPGA family. Furthermore, we want to thank David Oswald for many fruitful discussions and his advices on the Fourier preprocessing.

The work described in this paper has been supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II and by the German Federal Ministry of Education and Research BMBF (grant 01IS10026A, Project EXSET).

References

1. Side-channel Attack Standard Evaluation Board (SASEBO-GII). Further information is available via, http://staff.aist.go.jp/akashi.satoh/SASEBO/en/board/sasebo-g2.html
2. Braeken, A., Kubera, S., Trouillez, F., Touhafi, A., Mentens, N., Vliegen, J.: Secure FPGA Technologies and Techniques. In: FPL 2009, pp. 560–563. IEEE (2009)
3. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
4. Drimer, S.: Security for volatile FPGAs. PhD thesis, Computer Laboratory, University of Cambridge, United Kingdom (2009)
5. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 203–220. Springer, Heidelberg (2008)
6. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
7. Kepa, K., Morgan, F., Kosciuszkiewicz, K., Braun, L., Hübner, M., Becker, J.: FPGA Analysis Tool: High-Level Flows for Low-Level Design Analysis in Reconfigurable Computing. In: Becker, J., Woods, R., Athanas, P., Morgan, F. (eds.) ARC 2009. LNCS, vol. 5453, pp. 62–73. Springer, Heidelberg (2009)

8. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
9. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of Power Analysis Attacks on Smartcards. In: USENIX Workshop on Smartcard, pp. 151–161. USENIX Association (1999)
10. Moradi, A., Barenghi, A., Kasper, T., Paar, C.: On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx Virtex-II FPGAs. In: CCS 2011, pp. 111–124. ACM (2011)
11. Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-Engineering a Cryptographic RFID Tag. In: USENIX Security Symposium, pp. 185–194. USENIX Association (2008)
12. Note, J.-B., Rannaud, É.: From the Bitstream to the Netlist. In: FPGA 2008, p. 264. ACM (2008)
13. NVidia. NVIDIA's Next Generation CUDA Compute Architecture: Fermi (2009), http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIAFermiComputeArchitectureWhitepaper.pdf
14. Nvidia. CUDA Developer Zone (Website) (2011), http://developer.nvidia.com/category/zone/cuda-zone
15. Oswald, D., Paar, C.: Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 207–222. Springer, Heidelberg (2011)
16. Trimberger, S.: Trusted design in FPGAs. In: DAC 2007, pp. 5–8. ACM (2007)
17. Xilinx, Inc. Virtex-II Pro and Virtex-II Pro X FPGA User Guide (2002), http://www.xilinx.com/support/documentation/user_guides/ug012.pdf
18. Xilinx, Inc. Application Note XAPP151 (v1.7), Virtex Series Configuration Architecture User Guide (2004), http://www.xilinx.com/support/documentation/application_notes/xapp151.pdf
19. Xilinx, Inc. Virtex-4 FF668 Evaluation Board (2004), http://www.xilinx.com/products/boards-and-kits/HW-AFX-FF668-400.htm, User Guide, http://www.xilinx.com/support/documentation/boards_and_kits/ug078.pdf
20. Xilinx, Inc. Virtex-4 FPGA Configuration User Guide (2004), http://www.xilinx.com/support/documentation/user_guides/ug071.pdf
21. Xilinx, Inc. Application Note XAPP138 (v2.8), Virtex FPGA Series Configuration and Readback (2005), http://www.xilinx.com/support/documentation/application_notes/xapp138.pdf
22. Xilinx, Inc. Virtex-5 FPGA Configuration User Guide (2006), http://www.xilinx.com/support/documentation/user_guides/ug191.pdf
23. Ziener, D., Assmus, S., Teich, J.: Identifying FPGA IP-Cores Based on Lookup Table Content Analysis. In: FPL 2006, pp. 1–6. IEEE (2006)

Power Analysis of Atmel CryptoMemory – Recovering Keys from Secure EEPROMs

Josep Balasch[1], Benedikt Gierlichs[1], Roel Verdult[2], Lejla Batina[1,2], and Ingrid Verbauwhede[1]

[1] K.U.Leuven ESAT/COSIC and IBBT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be
[2] Radboud University Nijmegen, ICIS/Digital Security Group
Heyendaalseweg 135, 6525 AJ, Nijmegen, The Netherlands
{rverdult,lejla}@cs.ru.nl

Abstract. Atmel CryptoMemory devices offer non-volatile memory with access control and authenticated encryption. They are used in commercial and military applications e.g. to prevent counterfeiting, to store secrets such as biometric data and cryptographic keys, and in electronic payment systems. Atmel advertises the devices as "*secure against all the most sophisticated attacks,* [...] *including physical attacks*". We developed a successful power analysis attack on the authentication step of CryptoMemory devices. Despite the physical security claims by Atmel we found that the devices are not protected against power analysis attacks, except for counters that limit the number of (failed) authentication attempts, and thus power traces, to at most three. We examined the handling of these counters and discovered a flaw that allows us to bypass them, and to obtain power traces from an unlimited number of failed authentication attempts. Our attacks need as few as 100 power traces to recover the secret 64-bit authentication keys. From measurements to full key extraction, the attacks can be carried out in less than 20 minutes on a standard laptop. Once the keys are known, an adversary can read protected contents, clone devices, and manipulate the memory at will, e.g. to set the balance of an electronic wallet. To our knowledge, this is the first power analysis attack on Atmel CryptoMemory products reported in the literature.

Keywords: Atmel CryptoMemory, power analysis.

1 Introduction

In the past years, many commercial devices with security functionalities such as the KeeLoq-based remote keyless entry systems [19,22], the contactless Mifare DESFire MF3ICD40 card [29], or the FPGA bitstream encryption used by Xilinx Virtex FPGAs [26], were shown to be susceptible to power analysis attacks, e.g. DPA [23].

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 19–34, 2012.

The first "*secure memories with authentication*" [11,12] manufactured by Atmel appeared in 1999 under the name SecureMemory. These devices are mainly formed by a piece of EEPROM memory with access control logic and a cryptographic unit. A valid authentication grants read/write permissions to the memory. Failed authentications are recorded by authentication attempt counters (AACs), effectively locking the device after four or more attempts. A newer version of Atmel secure memories, commonly known as CryptoMemory, was released in 2002 as part of the Atmel AT88SCxxxxC series [7]. The AT88SCxxxxCRF [9] family, referred to as CryptoRF, appeared in 2003 providing the same features as CryptoMemory but with a contactless RF interface.

At the core of all these devices lies a proprietary stream cipher with secret specification, that we refer to as Atmel cipher. This cipher is employed during the mutual authentication protocol, in which a host and a CryptoMemory device authenticate each other by proving knowledge of a 64-bit shared secret key. A session key resulting from the mutual authentication can be further used to provide authenticated encryption of the communication channel.

CryptoMemory's security features, low cost, and ease of deployment have allowed these devices to be widely used in plenty of commercial and military applications [10]. A typical example is the use of the AT88SCxxxxC series to store High Bandwidth Digital Content Protection (HDCP) keys in products such as NVIDIA graphics cards [28], Labgear digital satellite receivers [4], or the Microsoft Zune player [30]. CryptoMemory devices are also used to strengthen security in systems vulnerable to counterfeiting schemes. They are for instance used in printers and printer cartridges manufactured by Dell, Ricoh, Xerox, and Samsung [1]. Other potential applications include "*appliances with smart batteries, set top boxes, video game consoles, video game cartridges, PDAs, GPS, and any system with proprietary algorithms or secrets*" [13], and storage of "*biometric information*" [10]. In smart card form, CryptoMemory devices are mainly used in vendor-specific electronic payments, e.g. in laundromats or parkings [2]. Further applications recommended by the manufacturer include "*ID and access cards, health care cards, loyalty cards, internet kiosks, energy meters, and e-government*" [14].

Contribution. We show that the secret authentication keys used by a CryptoMemory device can be extracted using basic, well-understood power analysis techniques. We devise a practical, non-invasive approach to exploit a flaw in the handling of AACs and to collect any number of power traces corresponding to failed authentication attempts. We have fully implemented and tested our attack on CryptoMemory devices from Atmel development kits and real world products, both in smart card and packaged IC form. Our attacks need as few as 100 power measurements for key extraction and can be carried out, including trace collection, in less than 20 minutes on a standard laptop. We also discuss how to fix the flaw in a way that future revisions of this (or similar) products are not vulnerable to such practical attacks, while remaining backwards compatible[1].

[1] We have informed Atmel about the vulnerabilities found in this work prior to its publication.

Paper Organization. In Section 2 we provide background information about CryptoMemory devices and their advertised security. We summarize previous work and we briefly sketch their security mechanisms. In Section 3 we develop an attack path for DPA attacks, and we provide the details and results of our attacks in Section 4. We discuss the implications of our findings as well as potential countermeasures in Section 5. We briefly conclude in Section 6.

2 Background

CryptoMemory devices are available either in plastic IC packages or as smart cards. The former communicate via a synchronous Two-Wire serial Interface (TWI), while the latter use the asynchronous T=0 protocol standardized in ISO 7816-3 [3]. CryptoMemory devices provide from 1 Kbit to 256 Kbits of memory divided into several user zones that are individually configurable with different access control policies. A separate configuration zone, customizable during the device personalization phase, is used to store such policies. A set of security fuses can be blown after the personalization phase in order to lock the configuration zone. CryptoMemory offers a total of three different security policies:

a) In password mode, a host simply needs to provide a 24-bit password to gain access to the zone. This mode offers a limited level of security, as all exchanged data (including passwords) is transmitted in the clear, i.e. it is vulnerable to eavesdropping and replay attacks.

b) In authentication mode, up to four 64-bit keys (in the following denoted by k) can be set during the personalization phase as shared secrets between host and device. An 8-bit AAC associated to each key controls the number of failed authentication attempts. The protected user zone(s) become inaccessible once AAC is set to x00[2]. Data transmitted to/from protected memory in this mode is in the clear but replay attacks do not apply.

c) In encryption mode, the communication channel between host and device is protected by authenticated encryption. A 64-bit shared session key K_s that is updated after each run of the mutual authentication protocol is used. Entering encryption mode requires the device to be already in authentication mode.

Regardless of the security policy, memory contents in the user zones are stored in the clear, i.e. CryptoMemory does not encrypt the data during the storage process.

Security of CryptoMemory devices. Atmel claims that CryptoMemory devices "*can secure data against all the most sophisticated attacks, including algorithmic attacks, systematic attacks, and physical attacks.*" [6]. In particular, physical attacks are counteracted by the use of tamper-proof features including metal shield layers over the active circuitry, encrypted internal buses, high-security test procedures, and defenses against timing and power supply attacks (to be understood as active tampering attacks, e.g. fault injection via glitching).

[2] In the most restrictive mode the maximum number of authentication attempts is set to four. The decreasing values of AAC are (xFF,xEE,xCC,x88,x00). A correct authentication automatically restores the value of AAC to xFF.

CryptoMemory also provides anti-tearing functionalities, i.e. in the event of a power loss during an EEPROM writing cycle, data can be recovered at the next power-up and written to the intended address. This feature is however optional, and it needs to be requested by the host prior to a write operation. A typical scenario in which this mechanism enhances security is payment systems, e.g. a malicious customer could try to remove a CryptoMemory-based card from the terminal slot before a decreased balance has been written to memory.

Surprisingly, we could not find a single reference to countermeasures against power analysis attacks in Atmel's marketing material and technical documentation. However, given the effort made to protect against invasive probing attacks and non-invasive tampering attacks, it is reasonable to assume that also power analysis attacks were considered. After all, it is claimed that "*CryptoMemory is designed to keep contents secure, whether operating in a system or removed from the board and sitting in the hacker's lab.*" [5]. In our view, it is likely that Atmel relies on the secrecy of the Atmel cipher and the AACs as countermeasures against basic power analysis attacks.

Previous Work. As it has previously occurred with other products using proprietary cryptographic algorithms [16,20,27], the security of CryptoMemory devices took a hit in 2010 when Garcia et al. [21] reverse-engineered the Atmel cipher and the authentication protocol used in the CryptoMemory family. The authors also cryptanalyzed these mechanisms and showed that an adversary could recover CryptoMemory authentication keys in 2^{52} cipher ticks using 2640 eavesdropped keystream frames. Biryukov et al. recently proposed an improved attack [15] that requires 30 eavesdropped keystream frames and 2^{50} cipher ticks to recover authentication keys. Other attacks against systems using CryptoMemory are known [24,32], but they exploit weaknesses in poorly designed protocols and mistakes during deployment rather than vulnerabilities of CrypoMemory devices.

Atmel Cipher. In the following, and for the sake of completeness, we briefly sketch the main functionality of the Atmel cipher. For a more formal and complete specification we refer the reader to [21].

Figure 1 depicts the inner structure of the Atmel stream cipher. The cipher state s is an element of $\mathbb{F}_2^{117}$ composed by a total of 4 shift registers. We denote these elements as left register l, middle register m, right register r, and feedback register f. In particular:

1. left register: $l = (l_0, l_1, ..., l_6) \in (\mathbb{F}_2^5)^7$

2. middle register: $m = (m_0, m_1, ..., m_6) \in (\mathbb{F}_2^7)^7$

3. right register: $r = (r_0, r_1, ..., r_4) \in (\mathbb{F}_2^5)^5$

4. feedback register: $f = (f_0, f_1) \in (\mathbb{F}_2^4)^2$

At each tick, the cipher state $s = (l, m, r, f)$ is transformed into a successor state $s' = (l', m', r', f')$ going through an intermediate state $\hat{s} = (\hat{l}, \hat{m}, \hat{r}, \hat{f})$. During this whole process the cipher takes a single input $a \in \mathbb{F}_2^8$ and produces an output keystream nibble $output(s') \in \mathbb{F}_2^4$.

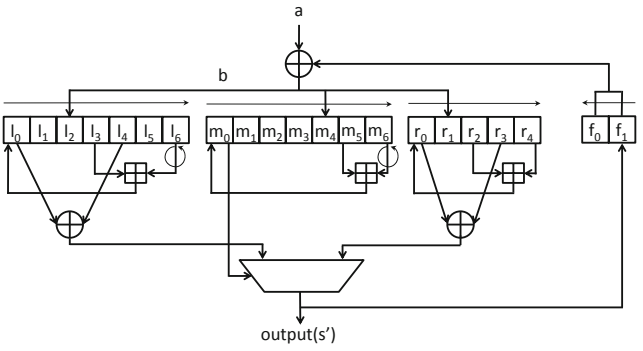


Fig. 1. Atmel cipher

Mutual Authentication Protocol. The mutual authentication protocol between a CryptoMemory device and a host is illustrated in Figure 2. Let $n_r \in (\mathbb{F}_2^8)^8$ be a host nonce, $n_t \in (\mathbb{F}_2^8)^8$ be a CryptoMemory device nonce, and $k \in (\mathbb{F}_2^8)^8$ a shared secret key. We denote a_r and $a'_r \in (\mathbb{F}_2^8)^8$ the host challenge authenticators, and a_t and $a'_t \in (\mathbb{F}_2^8)^7$ the device challenge authenticators. Both values are computed after feeding the values (n_r, n_t, k) into the Atmel cipher.

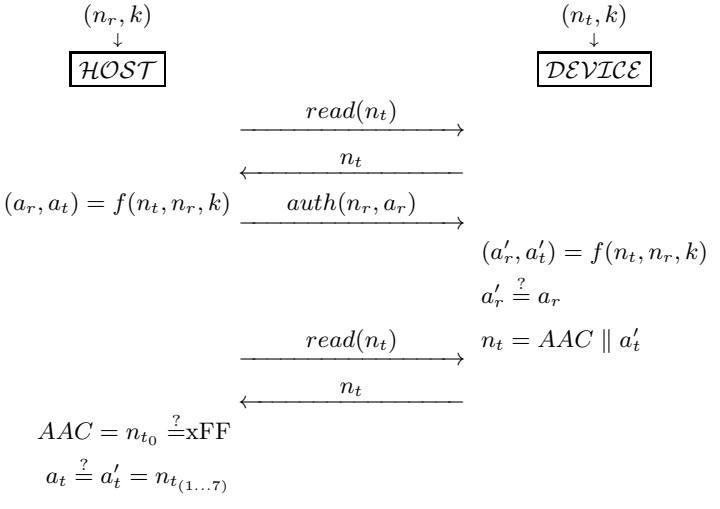


Fig. 2. CryptoMemory mutual authentication protocol

In the first phase of the protocol, the host reads the device randomness n_t and uses it, together with its own randomness n_r and the shared key k, to compute the authenticators (a_r, a_t). In the second phase, the host sends an authentication command to the device, namely $auth(n_r, a_r)$, including the value n_r and the first authenticator a_r. The device computes its own authenticators (a'_r, a'_t), and checks whether the provided a_r equals the calculated a'_r. If the check fails, the

State	Input bytes							Output nibbles						
(s_{0-6})	n_{t_0}	n_{t_0}	n_{t_0}	n_{t_1}	n_{t_1}	n_{t_1}	n_{r_0}	-	-	-	-	-	-	-
(s_{7-13})	n_{t_2}	n_{t_2}	n_{t_2}	n_{t_3}	n_{t_3}	n_{t_3}	n_{r_1}	-	-	-	-	-	-	-
(s_{14-20})	n_{t_4}	n_{t_4}	n_{t_4}	n_{t_5}	n_{t_5}	n_{t_5}	n_{r_2}	-	-	-	-	-	-	-
(s_{21-27})	n_{t_6}	n_{t_6}	n_{t_6}	n_{t_7}	n_{t_7}	n_{t_7}	n_{r_3}	-	-	-	-	-	-	-
(s_{28-34})	k_0	k_0	k_0	k_1	k_1	k_1	n_{r_4}	-	-	-	-	-	-	-
(s_{35-41})	k_2	k_2	k_2	k_3	k_3	k_3	n_{r_5}	-	-	-	-	-	-	-
(s_{42-48})	k_4	k_4	k_4	k_5	k_5	k_5	n_{r_6}	-	-	-	-	-	-	-
(s_{49-55})	k_6	k_6	k_6	k_7	k_7	k_7	n_{r_7}	-	-	-	-	-	-	-
(s_{56-62})	0	0	0	0	0	0	0	-	-	-	-	a_{r_0}	a_{r_1}	-
(s_{63-69})	0	0	0	0	0	0	0	-	-	-	-	a_{r_2}	a_{r_3}	-
(s_{70-76})	0	0	0	0	0	0	0	-	-	-	-	a_{r_4}	a_{r_5}	-
(s_{77-83})	0	0	0	0	0	0	0	-	-	-	-	a_{r_6}	a_{r_7}	-
(s_{84-90})	0	0	0	0	0	0	0	-	-	-	-	a_{r_8}	a_{r_9}	-
(s_{91-97})	0	0	0	0	0	0	0	-	-	-	-	$a_{r_{10}}$	$a_{r_{11}}$	-
(s_{98-104})	0	0	0	0	0	0	0	-	-	-	-	$a_{r_{12}}$	$a_{r_{13}}$	-
$(s_{105-111})$	0	0	0	0	0	0	0	-	-	-	-	$a_{r_{14}}$	$a_{r_{15}}$	-
$(s_{112-118})$	0	0	0	0	0	0	0	a_{t_0}	a_{t_1}	a_{t_2}	a_{t_3}	a_{t_4}	a_{t_5}	a_{t_6}
$(s_{119-125})$	0	0	0	0	0	0	0	a_{t_7}	a_{t_8}	a_{t_9}	$a_{t_{10}}$	$a_{t_{11}}$	$a_{t_{12}}$	$a_{t_{13}}$
$(s_{126-132})$	0	0	0	0	0	0	0	K_{s_0}	K_{s_1}	K_{s_2}	K_{s_3}	K_{s_4}	K_{s_5}	K_{s_6}
$(s_{133-139})$	0	0	0	0	0	0	0	K_{s_7}	K_{s_8}	K_{s_9}	$K_{s_{10}}$	$K_{s_{11}}$	$K_{s_{12}}$	$K_{s_{13}}$
$(s_{140-141})$	0	0						$K_{s_{14}}$	$K_{s_{15}}$					

Fig. 3. Generation of authenticators (a_r, a_t) and K_s given inputs (n_t, n_r, k)

value of AAC is decreased; otherwise, it is set to xFF. The device also updates the value of n_t by concatenating the 8-bit AAC with the 56-bit authenticator a'_t. In the final phase, the host reads the recently updated value of n_t. It first checks whether the authentication was successful (i.e. whether AAC holds the value xFF), and later compares the authenticator a_t with the provided a'_t. If all checks are correct, then the mutual authentication protocol succeeds.

The procedure to compute the authenticators (a_r, a_t) resulting of feeding the values (n_t, n_r, k) into the Atmel cipher is intuitively depicted in Figure 3. Each of the states s_i indicates one tick of the cipher for which an input byte a is given and an output nibble $output(s')$ is obtained. At the start of the protocol all registers of the internal state are initialized to zero. In a first phase, ranging from states s_0 to s_{55}, the three parameters (n_t, n_r, k) are scrambled into the cipher state. The output keystream nibbles generated by the cipher are for the moment ignored. In a second phase, from states s_{56} to s_{125}, all input bytes are set to zero. The output nibbles obtained after some of the ticks are used to form the authenticators a_r and a_t. In the final phase, the session key K_s is computed during states s_{126} to s_{141}.

3 Developing an Attack Path

Attack Goal. The aim of an adversary targeting CryptoMemory devices may vary depending on the deployment setting, but typically the objective will be

to either read protected information (e.g. to manufacture clone chips) or to overwrite memory contents (e.g. to increase the balance in payment scenarios).

In the following we will assume an adversary that possesses a CryptoMemory device configured either in authentication mode or encryption mode. Note that the first security operation in either of these modes is always the mutual authentication protocol. In other words, their security relies on the secrecy of the authentication key (or keys) k. If an attacker knew k, he could easily compute session keys K_s and encrypt/decrypt valid communications. Therefore, we define the goal of the adversary as the recovery of k.

Attack Approach. Previous work has already pointed out cryptographic weaknesses in the Atmel cipher, but the most efficient attack still requires substantial computational effort, e.g. two to six days computation on a cluster with 200 cores [15]. We focus on power analysis attacks as they are often more practical. Note that in the case of deployed CryptoMemory devices, eavesdropping the communication line for later cryptanalysis requires physical access to the target device. Thus, physical access for power analysis does *not* imply an additional requirement.

Target Devices. We have tested three different models of the CryptoMemory family, namely the chips AT88SC0404C, AT88SC1616C, and the newer AT88SC0808CA. These devices differ in the amount of user zone slots and/or size available to the end application, but the protocol to carry out the mutual authentication is identical for all of them.

Given that CryptoMemory devices do not contain a microcontroller, we can assume that the cryptographic unit, including the Atmel cipher, is fully implemented as a hardware module. We further assume that the implementation computes one cipher tick in one or two clock cycles, which gives us an idea of the type of pattern to look for in the power traces.

In the following we describe the experimental setup used in our analysis and the steps followed to evaluate the smart card versions of CryptoMemory. We summarize the slightly different approach required for CryptoMemory ICs, leading to the same results, at the end of the section.

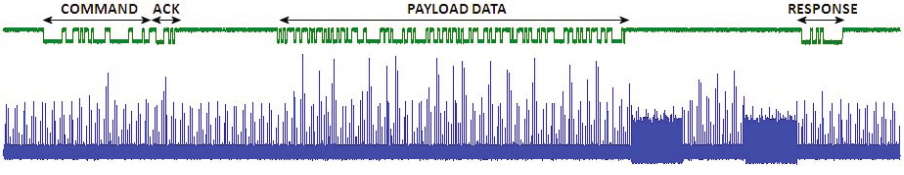
3.1 Experimental Setup

The main element of our experimental setup is a Virtex-II FPGA [33] that communicates with a PC (user interface) and a target device (CryptoMemory). The FPGA can be configured to communicate with either of the CryptoMemory device forms, i.e. as a T=0 compatible smart card reader or as a TWI master device. Apart from the communication line, the FPGA fully controls all external signals provided to the target device. For instance, the reset signal (RST), the external clock (CLK), and even the power supply (VCC) can be manipulated at will and at any point in time. A 50 Ohm resistor is inserted in series in the ground (GND) line of the target device. We use a Tektronix DPO7254 [31] oscilloscope to measure the voltage drop over this resistor, thus obtaining a trace proportional to the device's power consumption. Note that we do not exhaust

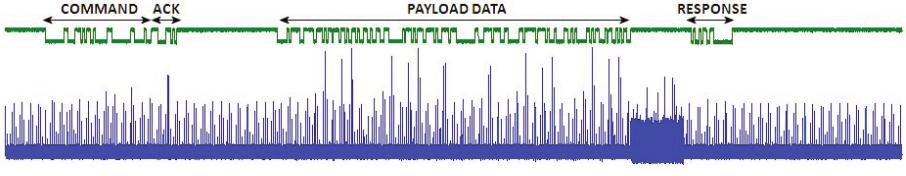
the capabilities of the oscilloscope and that a low-cost model could be used just as well. For all experiments we used a sampling rate of 100 MS/s and 20 MHz bandwidth. In addition to the power consumption we also use the oscilloscope to monitor RST, CLK, and I/O lines.

3.2 Initial Investigation of Power Traces

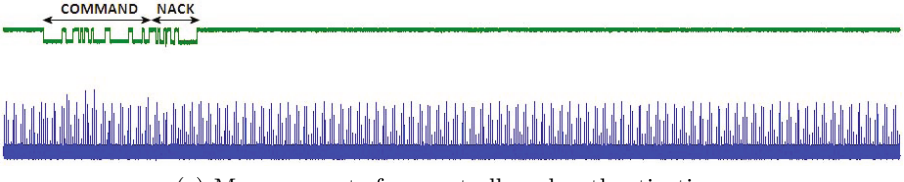
The first step in a power analysis attack is to determine in which part of the protocol the secret key is used by the target device, i.e. at which points in time are the key bytes fed as input to the Atmel cipher. An overview of the I/O and a power trace obtained during the processing of a mutual authentication command is shown in Figure 4. Figure 4(a) represents a successful authentication, Figure 4(b) represents a failed authentication, and Figure 4(c) represents an authentication attempt when the AAC is set to x00.



(a) Measurements for a successful authentication.



(b) Measurements for a failed authentication.



(c) Measurements for a not allowed authentication.

Fig. 4. I/O and power trace for a successful authentication (a), a failed authentication (b), and a not allowed authentication (c)

The leftmost part of the figures start with the host sending 5 bytes corresponding to an authentication command. After receiving these bytes, the card can reply either with an acknowledgement ACK, indicating that it accepts the command, or with a negative acknowledgement NACK, refusing to process the command. The latter is shown in Figure 4(c), where the CryptoMemory device has locked access to the associated user zones. If the device acknowledges the

command then the host sends the payload data, i.e. 16 bytes corresponding to the values n_r and a_r. Upon reception, the card performs a series of operations to determine whether the authenticator provided by the host is correct. Finally, the card sends a response to the host indicating the outcome of the authentication attempt.

The most interesting part of the figure corresponds to the card calculations upon reception of the payload data. One can clearly notice that the calculation interval in Figure 4(b) is significantly shorter than in Figure 4(a). Further, a clearly distinguishable pattern with higher power consumption appears twice for the valid authentication and only once for the invalid authentication. This pattern corresponds to EEPROM writings in configuration memory: the first one, present for both valid and invalid attempts, corresponds to decreasing the value of AAC; the second one, only present for the valid authentication, corresponds to writing the new value of n_t and the session key K_s. Note that writing a new value n_t implies a valid authentication, and thus AAC is restored back to xFF.

Although not visible in these overview plots, we noticed that the cryptographic unit appears to run at a clock frequency slower than that we provided externally. Further investigations showed that the device internally derives this slower clock signal by dividing the external clock by approximately a factor 200.

The card's calculation of the values (a'_r, a'_t) must happen before the second EEPROM write operation, simply because the n_t cannot be updated without having computed a'_t beforehand. Our experiments have shown that the device computes the authenticators (a'_r, a'_t) on-the-fly, i.e. while the payload data (n_r, a_r) is being received from the host. Similarly, the calculation of the session key K_s is performed between the two EEPROM writings, only after the device has authenticated the host by verifying that a_r and a'_r are equal.

Figure 5 shows a zoomed version of the I/O and power traces during the transmission of the value $n_r = (n_{r_0}, \ldots, n_{r_7})$ to the card. A clearly distinguishable peak in the power trace is visible at the end of each byte transmission, while a total of six high peaks are also identifiable during the transmission of a byte. As explained in Section 2 the calculation of (a'_r, a'_t) requires 126 cipher ticks. The pattern in the power traces has a perfect mapping with the cipher behavior illustrated in Figure 3 considering a hardware implementation. In fact, the first peak in Figure 5 corresponds to the state s_6, i.e. when the value n_{r_0} is fed to the cipher. The following six peaks correspond to states s_7 to s_{12}, in which the card uses its own randomness (values n_{t_2} and n_{t_3}) as inputs. Once n_{r_1} has been received over the I/O line, a new peak corresponding to state s_{13} appears in the power trace, and the pattern is repeated for every transmitted byte. In summary, each of the 50 peaks highlighted in Figure 5 corresponds to a cipher state ranging from s_6 to s_{55} in Figure 3.

Since the key k is scrambled into the cipher states s_6 to s_{55}, the device might leak sensitive information through its power consumption. We could not immediately identify a visible pattern in the power consumption that could relate to k (SPA leak) but we also did not expect that from a hardware implementation of a stream cipher. Therefore, we focused our attention on attacks that use statistical

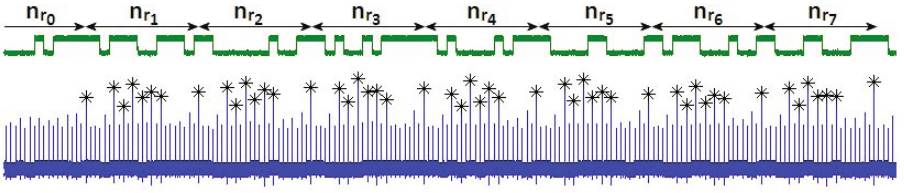


Fig. 5. I/O and power traces during the transmission of n_r in authentication command. Interesting peaks are marked with *.

post-processing of the collected power traces. Recall that DPA attacks require the processing of multiple power traces corresponding to multiple authentication attempts. For each attempt the device must use the same (unknown) k and varying input data.

3.3 Overcoming Authentication Attempt Counters

Even though we have identified the parts in the power traces that correspond to processing of the secret key k, an important practical issue still remains. In our adversarial model, the adversary possesses a CryptoMemory device that is *already* configured. Thus, he does not know the secret key k. In order to run the mutual authentication protocol, an adversary needs to provide an authenticator a_r to the device. However, as the attacker cannot compute this value correctly, the CryptoMemory device will not authenticate the host and, as a consequence, it will decrease the associated AAC. Given that the user zones become inaccessible to the host after four failed authentication attempts, an adversary can collect at most three power traces before permanently locking the device. The issue is that three traces are clearly not sufficient to carry out a successful DPA attack.

There are several ways to try to deal with this limitation. If the application under attack were to use the same key in all deployed CryptoMemory devices, then it would suffice to collect power measurements from several devices. One could also try to take many measurements from a single device, effectively sacrificing it. But as can be seen in Figure 4(c), once the AAC value is set to x00 the device no longer computes the authenticators and measurements would thus be worthless. Therefore, an adversary could obtain at most four power traces per device tested. However, a scenario in which all devices share a single key k seems unlikely to be found in secure deployments[3].

An alternative could be to use template attacks as introduced by Chari et al. [18]. The devices provided by Atmel in evaluation kits, completely configurable by the user, could be used to build such templates. We expect template attacks to require very few power traces from the target device, but it is not clear if three traces would suffice, due to the large cipher state. We did not investigate this approach further as we were interested in more simple attack paths.

[3] Atmel actually recommends to diversify keys in all CryptoMemory deployments by combining the configurable device ID with a unique master key [8], e.g. using a hash function.

We followed a more intuitive approach to overcome the limitation imposed by the AACs. Recall from Figure 4 that *all* the information required to perform DPA, i.e. key-dependent information in power measurements, is obtained *while* the value n_r is being sent to the card. In other words, the information is available to the attacker *before* the card actually decreases the value of AAC. Our approach consists in injecting a negative pulse in the RST signal of the device before the new value of AAC is written into configuration memory. Doing so forces the device to reset its state before beginning the EEPROM write operation.

A successful implementation of this simple procedure is shown in Figure 6. Besides the I/O and power traces, the figure also shows the RST signal input to the device. Injecting a pulse on the RST line right after sending the payload data successfully resets the device. This can be observed on the I/O line, as the device sends its Answer-To-Reset (ATR) value to the host device right after the rising edge in RST. Note also that the first EEPROM write pattern indicating AAC being decreased does not appear in the power trace.

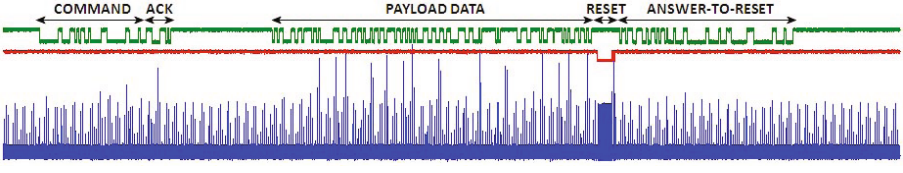


Fig. 6. I/O, RST, and power traces during interrupted authentication

The timing of the RST pulse is far from critical as the adversary can inject it at any point after the reception of n_r and before the first EEPROM writing, i.e. during the transmission of the value a_r. Recall that the transmission of a bit in the I/O line requires 372 clock cycles, and a byte transmission requires a total of 12 bits (1 start bit, 8 data bits, 1 parity bit, and 2 stop bits) [3]. Taking that into account, the adversary has an interval of 35 712 clock cycles in which the RST pulse can be sent to the card.

Note that resetting the device before the EEPROM writings implies that the value n_t can never be updated. As a consequence, all power measurements collected using this approach will correspond to the same device randomness n_t. Although in other scenarios this characteristic could be problematic, in our case it does not limit the success or applicability of DPA attacks. This is because an attacker can still provide varying values of n_r for each authentication attempt, which is fed into the Atmel cipher some ticks before the key k.

Differences with CryptoMemory Packaged ICs. Besides some particularities caused by the use of TWI instead of T=0, the overall behavior of CryptoMemory packaged ICs resembles what we have presented until now. Most important is the fact that the calculation of the parameters (a'_r, a'_t) is done in exactly the same way as in the smart card, i.e. on-the-fly while the host sends the values (n_r, a_r). It is thus possible to identify which zones of the power

measurements correspond to which states of the Atmel cipher during the feeding of the values (n_t, n_r, k).

The main physical difference between CryptoMemory packaged ICs and smart cards is that the former do not have an external RST pin. This could have been a problem, but an "equivalent" mechanism to overcome the AACs consists in cutting the supply voltage (VCC) before the counters are decreased. Similarly to the RST mechanism the timing accuracy to cut the voltage is not critical, and the adversary has plenty of time to perform it[4].

Once the power traces are obtained, the attacks on CryptoMemory in smart card form and in IC form are identical and lead to very similar results.

4 Power Analysis Attack

We obtained a set of 1000 power traces sampled during executions of the mutual authentication protocol, for which we provided random nonces n_r to the device and, each time, reset it as described above.

We processed the traces with a simple routine that extracts the peaks highlighted in Figure 5, yielding a new set of highly compressed and well aligned traces. Contrary to typical attacks on block cipher implementations, the key bytes can not be recovered independently but should be recovered in the order in which they are fed into the Atmel cipher implementation, i.e. first k_0, then k_1, etc.

We first investigated the feasibility of basic DPA attacks that recover k one byte at a time. We used a Hamming distance power model on the full cipher state, i.e. the total number of bit flips in the transition from state s to state s', and Pearson's correlation coefficient as distinguisher [17].

As explained in Section 2, each key byte is fed into the cipher three times in consecutive cipher ticks. Our basic attack worked best when we attacked the last cipher tick that fed in a given key byte, e.g. for byte k_0 the transition from state s_{29} to s_{30}.

Figure 7 exemplarily shows the results we obtained when attacking k_6 (the worst case). The left part of the figure shows the correlation coefficients for all 256 hypotheses, plotted over "time", computed using all 1000 measurements. The right part of the figure shows the evolution of the maximum and minimum correlation peaks per key hypothesis, plotted over the number of traces used. We verified that all key bytes can be recovered in this way using less than 500 traces.

Our slightly more elaborate attack additionally exploits two simple facts. First, we know exactly which sample in the power traces corresponds to which cipher tick. Thus, the attack focuses on the correct sample in the traces and ignores all other samples that appear as noise. Second, each key byte is fed into

[4] CryptoMemory packaged ICs require the host to perform acknowledge polling after sending a mutual authentication command. If the host does not send any polling command the IC is effectively idle, which gives enough room for an adversary to switch off the supply voltage.

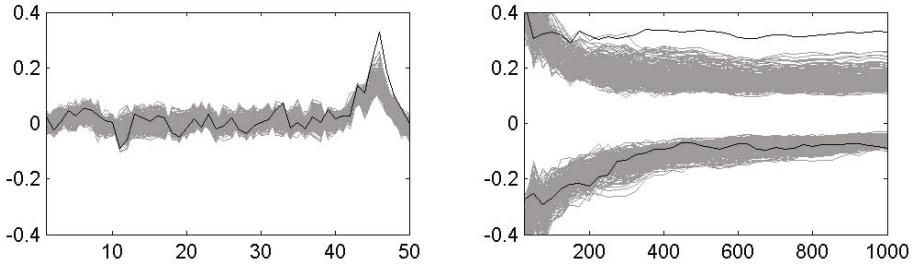


Fig. 7. Results of basic DPA attack. Correlation coefficients per key hypothesis (left), and evolution of correlation peaks per key hypothesis (right).

the cipher three times. Thus, the attack targets all three transitions and adds up the obtained correlation coefficients, per key hypothesis.

In addition, the attack further exploits that the dependence of the cipher state on k grows only slowly and byte per byte. Similar to the strategy described in [19], the attack does not aim to immediately identify the exact value of a key byte, but it maintains a list of the best candidate values and re-evaluates them when attacking the next key bytes.

We verified that this enhanced attack recovers the correct key k using less than 100 measurements. We note that both attacks, from measurements to full key recovery, can be carried out in less than 20 minutes on a standard laptop. Algebraic side-channel analysis would perhaps allow to work with even fewer traces, but it requires to build templates.

5 Implications and Countermeasures

We have shown that an adversary can easily extract the secret authentication key(s) k from CryptoMemory devices using basic, well-understood power analysis techniques. As a consequence, the adversary can perform all actions that any authenticated host could perform. This includes reading the memory, which allows to clone the device, and manipulating its memory contents at will.

The success of our attack is due to two design flaws in CryptoMemory. First, the implementation of the Atmel cipher is not protected by countermeasures against power analysis attacks, except for the AACs that limit the number of traces that can be obtained from a device before it locks itself to at most three. And second, an inadequate handling of the AACs that allows an adversary to bypass this limitation and to obtain any number of measurements from a given device without locking it.

A simple way to prevent our attack would be to modify the handling of the AACs. As learned from our experiments, CryptoMemory performs the authentication procedure as follows: compute the authenticators, decrease the value of AAC, compare the authenticators, and update the value of AAC according to success/failure when writing n_t in memory. This sequence can also be extracted

from Figures 4(a) and 4(b). Atmel explicitely states that this procedure is used "*to prevent attacks*" [7]. In fact, the method protects only the comparison operation and is often used e.g. in SIM cards during PIN verification. Our attacks do, however, not target the comparison operation but the time instants when the secrets are manipulated. Since CryptoMemory manipulates the secret key(s) *before* it decreases AAC, the counters can be easily bypassed with a reset. A more secure handling of the AACs could be to decrease the counter right upon reception of a mutual authentication command, and *prior* to the reception of the payload data and computation of the authenticators.

Protecting against more sophisticated attacks than ours may require to implement some of the well-known countermeasures against power analysis attacks, e.g. noise generators and power filters at the hardware level, or masking, random delays, and shuffling at the circuit level [25].

6 Conclusions

CryptoMemory is advertised to be used as a secure element in larger systems, in a way that only authorized hosts with knowledge of the correct authentication keys can have access to the protected memory contents. It is ensured that CryptoMemory "*permanently lock these values* [secret keys] *after device personalization and guarantee these values can never be read*" [8].

In this work we have shown how to extract such keys by using well-understood power analysis techniques. CryptoMemory uses fuses to lock access control policies, access control to protect memory contents, and AACs to strengthen access control. Therefore, a large part of CryptoMemory's security relies on the AACs, that we identified as a not sufficiently protected point of failure.

Acknowledgments. This work was supported in part by the European Commissions ECRYPT II NoE (ICT-2007-216676), by the Belgian States IAP program P6/26 BCRYPT, and by the Research Council K.U. Leuven: GOA TENSE (GOA/11/007). Josep Balasch and Roel Verdult are funded by a PhD grant within the covenant between the universities K.U. Leuven and R.U. Nijmegen. Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

1. AT88SC0204 ChipResetter, http://chipreset.atw.hu/6/index61.html
2. Coinamatic, http://www.coinamatic.com
3. ISO/IEC 7816-3: Identification cards - integrated circuit(s) cards with contacts - part 3: Electronic signals and transmission protocols (1997)
4. Labgear HDSR300 High Definition Satellite Receiver. User Guide, http://www.free-instruction-manuals.com/pdf/p4789564.pdf
5. Anderson, D.: Understanding CryptoMemory - The World's Only Secure Serial EEPROM, http://www.atmel.com/atmel/acrobat/doc5064.pdf

6. Atmel. CryptoMemory features, http://www.atmel.com/microsite_cryptomemory/iwe/index.html?source=tout_other2
7. Atmel. CryptoMemory Specification, http://www.atmel.com/atmel/acrobat/doc5211.pdf
8. Atmel. CryptoMemory Powerful Security at Low Cost, http://www.atmel.com/atmel/acrobat/doc5259.pdf
9. Atmel. CryptoRF Specification, http://www.atmel.com/atmel/acrobat/doc5276.pdf
10. Atmel. News Release, http://www.cryptomemorykey.com/pdfs/AtmelPR.pdf
11. Atmel. Secure Memory with Authentication AT88SC153, http://www.atmel.com/atmel/acrobat/doc1016.pdf
12. Atmel. Secure Memory with Authentication AT88SC1608, http://www.atmel.com/atmel/acrobat/doc0971.pdf
13. Atmel Corporation. Plug-and-Play Crypto Chip for Host-Side Security, http://www.atmel.com/dyn/corporate/view_detail.asp?ref=&FileName=Cryptocompanion_2_26.html&SEC_NAME=Product
14. Benhammou, J.P., Jarboe, M.: Security at an affordable price. Atmel Applications Journal, 29–30 (2004)
15. Biryukov, A., Kizhvatov, I., Zhang, B.: Cryptanalysis of the Atmel Cipher in SecureMemory, CryptoMemory and CryptoRF. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 91–109. Springer, Heidelberg (2011)
16. Bogdanov, A.: Linear Slide Attacks on the KeeLoq Block Cipher. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 66–80. Springer, Heidelberg (2008)
17. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
18. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
19. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 203–220. Springer, Heidelberg (2008)
20. Garcia, F.D., de Koning Gans, G., Muijrers, R., van Rossum, P., Verdult, R., Schreur, R.W., Jacobs, B.: Dismantling MIFARE Classic. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 97–114. Springer, Heidelberg (2008)
21. Garcia, F.D., van Rossum, P., Verdult, R., Schreur, R.W.: Dismantling SecureMemory, CryptoMemory and CryptoRF. In: Keromytis, A., Shmatikov, V. (eds.) Proceedings of ACM CCS 2010, pp. 250–259. ACM Press (2010)
22. Kasper, M., Kasper, T., Moradi, A., Paar, C.: Breaking KeeLoq in a Flash: On Extracting Keys at Lightning Speed. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 403–420. Springer, Heidelberg (2009)
23. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
24. Lee, J., Pahl, N.: Bypassing Smart-Card Authentication and Blocking Debiting: Vulnerabilities in Atmel CryptoMemory based Stored-Value Systems. DEFCON 18 (2010)
25. Messerges, T.: Power analysis attack countermeasures and their weaknesses. In: CEPS Workshop (2000)

26. Moradi, A., Barenghi, A., Kasper, T., Paar, C.: On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks Extracting Keys from Xilinx Virtex-II FPGAs. In: Danezis, G., Shmatikov, V. (eds.) Proceedings of ACM CCS 2011, pp. 111–124. ACM Press (2011)
27. Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-engineering a cryptographic RFID tag. In: Proceedings of USENIX 2008, pp. 185–193. USENIX Association (2008)
28. NVIDIA. Checklist for Building a PC that Plays HD DVD or Blu-ray Movies, ftp://download.nvidia.com/downloads/pvzone/Checklist_for_Building_a_HDPC.pdf
29. Oswald, D., Paar, C.: Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 207–222. Springer, Heidelberg (2011)
30. Hearst Electronic Products. Microsoft Zune HD 16GB, what's inside, http://www2.electronicproducts.com/Microsoft_Zune_HD_16GB-whatsinside_text-89.aspx
31. Tektronix. DPO7000C Oscilloscope Series, http://www.tek.com/products/oscilloscopes/dpo7000/
32. Viksler, H.: Web Laundry (In)Security, http://ihackiam.blogspot.com/2010/09/web-laundry-insecurity.html
33. Xilinx. XUP Virtex-II Pro Development System User Manual, http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf

Short Transitive Signatures for Directed Trees

Philippe Camacho and Alejandro Hevia

Dept. of Computer Science, University of Chile,
Blanco Encalada 2120, 3er piso, Santiago, Chile
{pcamacho,ahevia}@dcc.uchile.cl

Abstract. A transitive signature scheme allows us to sign a graph in such a way that, given signatures on edges (a, b) and (b, c), it is possible to compute the signature on edge (a, c) without the signer's secret. Constructions for undirected graphs are known but the case of directed graphs remains open. A first solution for the particular case of directed trees ($DTTS$) was given by Yi at CT-RSA 2007. In Yi's construction, the signature for an edge is $O(n \log(n \log n))$ bits long in the worst case where n is the number of nodes. A year later in Theoretical Computer Science 396, Neven proposed a simpler scheme where the signature size is reduced to $O(n \log n)$ bits. Although this construction is more efficient, $O(n \log n)$-bit long signatures still remain impractical for large n.

In this work, we propose a new $DTTS$ scheme such that, for any value $\lambda \geq 1$ and security parameter κ: (a) edge signatures are only $O(\kappa\lambda)$ bits long, (b) signing or verifying an edge signature requires $O(\lambda)$ cryptographic operations, and (c) computing (without the secret key) an edge signature in the transitive closure of the tree requires $O(\lambda n^{1/\lambda})$ cryptographic operations. To the best of our knowledge this is the first construction with such a trade off.

Our construction relies on *hashing with common-prefix proofs*, a new variant of collision resistance hashing. A family $\mathcal{H}$ provides hashing with common-prefix proofs if for any $H \in \mathcal{H}$, given two strings X and Y equal up to position i, a prover can convince anyone that $X[1..i]$ is a prefix of Y by sending only $H(X), H(Y)$, and a small proof. We believe that this new primitive will lead to other interesting applications.

Keywords: Transitive Signatures, Collision-Resistant Hashing.

1 Introduction

Transitive signatures is a primitive introduced by Micali and Rivest [14] where a signer wants to authenticate a graph. The main property of such scheme is that, given the signatures of edges (a, b) and (b, c), it is possible to compute - *without the knowledge of the secret* - a signature for the edge (a, c). In their work, the authors propose an efficient scheme for undirected graphs based on the difficulty of computing discrete logarithm for large groups. They left the existence of a transitive signature scheme for directed graph (DTS) as a challenging open question.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 35–50, 2012.

The easier problem of transitive signatures for directed trees ($DTTS$) was first addressed by Yi [19]. Solutions for this case, even though it is a special kind of directed graph, are still interesting in practice. For example they allow to implement efficient *military chains of command* where the presence of a path between a and b means b must execute orders of a. Yi's construction, based on a special assumption for the RSA cryptosystem, yields signatures of size $O(n \log(n \log n))$ bits, where n is the number of nodes of the tree. Neven [16] described a simpler solution based only on the existence of standard digital signatures which also reduces signature size to $O(n \log n)$ bits.

In this work, we describe a new construction for a $DTTS$ scheme that enjoys much better worst-case complexity. We obtain a scheme where, for any $\lambda \geq 1$, (a) signing or verifying an edge signature requires $O(\lambda)$ cryptographic operations, and (b) computing (without the secret key) an edge signature in the transitive closure of the tree requires $O(\lambda n^{1/\lambda})$ cryptographic operations. Also, signature size is substantially improved: our signatures are only $O(\kappa\lambda)$ bits, where κ is the security parameter. In particular, if $\lambda = \log(n)$ then signatures are only $O(\kappa \log(n))$ bits, while allowing efficient signature computation ($O(\log(n)$ time). Alternatively, by setting for example $\lambda = 2$, we obtain optimal signature sizes of $O(2 \cdot \kappa) = O(\kappa)$ bits if we are willing to afford $O(\sqrt{n})$ computation.

Our Approach. There are two main ideas in our construction. First, we use the following fact observed by Dietz [9]. Let `Pre` and `Post` be the two strings representing the sequences of nodes obtained by pre-order and post-order (respectively) traversal of a given tree $\mathcal{T}$. Dietz observed that there exists a path from a to b if and only if a appears before b in `Pre` and b appears before a in `Post`. This property captures the fact that the tree is directed (from top to bottom) and gives us a characterization of the existence of a path between two nodes. Armed with this result, we reduce the problem of deciding whether there is a path between vertices a and b to comparing the position of a and b in a string sequence $\mathcal{S}$. Doing this efficiently is not trivial as the tree can grow, which means the string $\mathcal{S}$ dynamically changes. An *order data structure* – a concept also introduced in [9] – does exactly what we need: it supports element insertions into a sequence while still providing an efficient method to decide element order. Roughly speaking, we implement such data structure via a binary search tree $\mathcal{B}$, where each pair of elements x and y in $\mathcal{S}$ are associated to nodes $x, y \in \mathcal{B}$ (respectively), each with efficiently computable short labels $\ell(x)$ and $\ell(y)$. We then are able to define the relation "a appears before b in the sequence $\mathcal{S}$" as a total order relation $\prec$ which can be efficiently evaluated only from $\ell(a)$ and $\ell(b)$.

To achieve this, we use a labelling technique – based on *tries* [10] – which allows efficient and *incremental* computations of labels for new elements. Any newly inserted element v in $\mathcal{T}$ is mapped to a node v in $\mathcal{B}$ whose label $\ell(v)$ will share all but the last bit with another already computed label (see details in Section 4). Thus, whether an element a comes before some other element b in $\mathcal{S}$ can then be efficiently tested by lexicographical comparison between the labels associated to the corresponding nodes in $\mathcal{B}$. With this at hand, we then use *two*

of these data structures to keep track of `Pre` and `Post` lists and to test Dietz's condition on any pair of elements.

The problem, however, is that labels of $O(n)$ bits are now associated to vertices of the n-node tree $\mathcal{T}$, so at first sight little has been gained: signature length is now $O(n)$ bits compared to $O(n \log n)$ bits in Neven's construction. That is when our second idea comes into play: We use a new kind of collision-resistant hash function with an extra property. Given only two hash values $H(A)$, $H(B)$ and a small proof one can convince a *verifier* that A and B share a common prefix up to a position i. We call this new primitive *hashing with common-prefix proofs*. This tool allows us to prove Dietz's condition using only *hashed* labels (and a constant size proof), effectively compressing the signature.

We further improve our construction by showing how to balance the work between the *verifier* and the *combiner* (the participant who *combines* edge signature without the help of the *signer*) using the natural idea of hashing consecutive chunks of the initial string to obtain a shorter one, and repeat this operation several times. This technique leads to a novel tradeoff $O(\lambda n^{1/\lambda})$ vs. $O(\lambda)$ for $\lambda \geq 1$ between the time to compute a proof versus the time to verify a proof. The security of our primitive is based on the *q-Bilinear Diffie Hellman Inversion* assumption, introduced by Boneh and Boyen [3].

Related Work. The concept of transitive signatures was introduced by Rivest and Micali [14] who also gave constructions for undirected graphs. Bellare and Neven in [2], as well as Shahandashti et al. in [17], introduced new schemes based on bilinear maps (but still for undirected graphs). Hohenberger [12] showed that the existence DTS implies the existence of abelian groups where inversion is computationally infeasible except when given a trapdoor. Such groups are not known to exist either. Transitive signatures are a special case of homomorphic signatures, a primitive introduced by Rivest and explored in [13,5,4]. In [18], a stateless $DTTS$ scheme with constant size signature (as opposed to ours which is constant size but stateful) is proposed but without security proof. Finally, we observe that using accumulators techniques [7,8] we can improve Neven's construction [16] in order to obtain short signatures. Such a solution, however, does not provide two key properties we achieve in this work. In particular, we want the computation of edge signatures to be paralellizable, and a scheme that allows unbounded growth for the trees (our construction is able to increase the bound on the number of nodes by dynamically updating setup parameters, see Sect. 3). We explore a $DTTS$ construction based on accumulators in the full version [6].

Our contributions. First, we propose a new practical $DTTS$ scheme which, to the best of our knowledge, is the most efficient one to the date. Our scheme also provides a flexible tradeoff between signature computation and verification. To achieve this goal, we describe a general and practical new type of collision-resistant hashing called hashing with common-prefix proofs ($HCPP$). $HCPP$ functions enable efficient proofs that certain strings share common-prefixes.

We believe that this primitive may lead to many applications in the field of authenticated data structures.

Organization of the paper. In Section 2, we introduce notations, definitions for $DTTS$, and the complexity assumptions we use. Section 3 describes our new primitive in detail. Then in Section 4, we show how to use *hashing with common-prefix proofs* to obtain a practical $DTTS$ scheme. We conclude in Section 5.

2 Preliminaries

Notations and Conventions. If $\kappa \in \mathbb{N}$ is the security parameter then 1^κ denotes the unary string with κ ones. A function $\nu : \mathbb{N} \to [0,1]$ is said to be negligible in κ if for every polynomial $p(\cdot)$ there exists κ_0 such that $\forall \kappa > \kappa_0 : \nu(\kappa) < 1/p(\kappa)$. In the following, neg will denote *some* negligible function in κ. An algorithm is called PPT if it is probabilistic and runs in polynomial time in κ. We write $x \xleftarrow{R} X$ to denote an element x chosen uniformly at random from a set X. Finally, in the rest of this work, we use the convention that time complexities are expressed in terms of the number of cryptographic operations (signatures, group exponentiations, and bilinear map computations).

Strings. A string S of size $m = |S|$ is a sequence of symbols $S[1], S[2], ..., S[m]$ from some alphabet Σ. We assume there is a total order relation $<$ over Σ. If $m = 0$ then $S = \epsilon$ is the *empty* string. $S[i..j]$ denotes the substring of S starting at position i and ending at position j (both $S[i]$ and $S[j]$ are included). In particular if $A = S[1..j]$ for some $j \geq 0$ then we say that A is a prefix of S (by convention $A[1..0]$ for any string A is the empty string ϵ). The concatenation operator on strings is denoted as $||$. We say a string C is a common prefix of A and B if C is prefix of A and also of B. String C is said to be the *longest common prefix* of A and B if C is a common-prefix of A and B but $C||\sigma$ is not a common prefix of A and B for any symbol $\sigma \in \Sigma$. Without lost of generality, we assume $<$ is the (standard) lexicographical order on $\{0,1\}^*$. We single out \$ as a special symbol that is used only to mark the end of a string, and satisfies $0 < \$ < 1$. We define the *extended* lexicographical order $\prec$ on $\{0,1\}^*$ as following: let $X, Y \in \{0,1\}^*$ and $X' = X||\$, Y' = Y||\$$ strings obtained by appending the end marker to X, Y. We say that $X \prec Y \Leftrightarrow X' < Y'$.

Trees. Let $\mathcal{T} = (V, E)$ be a directed tree. The transitive closure of $\mathcal{T}$ is $\mathcal{T}^* = \{(a,b) : a, b \in \mathcal{T}$ and there is a path from a to $b\}$. When considering depth-first traversals of a tree, we denote by `Pre` and `Post` the strings formed by concatenating the successive labels of the nodes visited in a *pre-order* (i.e. the node is appended to the string when it is visited for the first time) respectively *post-order* (i.e. the node is appended to the string when it is visited for the last time). Our construction also makes use of binary *tries* [10], a type of binary tree, which associate labels to each node as follows. First, for each node, the edge going to a left (resp. right) child is tagged 0 (resp. 1). Then, the label for the node is obtained by concatenating the tags on the edges in a path from the

root to the node. This way, any node x in the trie $\mathcal{B}$ can be identified by its associated label $X \in \{0,1\}^*$. Given some label X, we denote by $node(X)$ the corresponding node in $\mathcal{B}$ if it exists. We say a node $x' \in \mathcal{B}$ is a descendant of x if x' belongs to the sub-tree rooted at x or equivalently if there is a path from x to x'. The lowest common ancestor of two nodes x, y of $\mathcal{B}$ is the node z such that x and y belong to the sub-tree rooted at z, and for any child z' of z, x or y is not a descendant of z'.

Collision Resistance and Standard Signatures Schemes. The family of functions $\mathcal{H}$ is said to be collision-resistant ($CRHF$) if, for $H\colon \{0,1\}^* \to \{0,1\}^\kappa$ uniformly chosen at random in $\mathcal{H}$, any computationally bounded adversary can not find two different messages M and M' such that $H(M) = H(M')$ except with negligible probability. Let $Alg_H(\cdot)$ be a PPT algorithm that computes H, then if $Alg_H(\cdot)$ is fed with input X and returns y, we write $X = H^{-1}(y)$. We denote by $\mathsf{SSig} = (\mathsf{SKG}, \mathsf{SSig}, \mathsf{SVf})$ a standard signature scheme. A pair of private/public keys is created by running $\mathsf{SKG}(1^\kappa)$. Given a message $M \in \{0,1\}^*$, a signature on M under pk is $\sigma_M = \mathsf{SSig}(sk, M)$. A signature σ on M is deemed valid if and only if $\mathsf{SVf}(pk, M, \sigma)$ returns valid. Regarding security, we use the standard notion of existential unforgeability under chosen message attack [11].

Transitive Signatures for Directed Trees. In a *transitive* signature scheme for directed trees, the *signer* can dynamically sign edges in a directed tree. Then without the secret, given two signed edges (a, b) and (b, c) it is possible to *combine* them into a signature on the edge (a, c). This property enables the computation of signature on any path in the tree.

Definition 1. *(Transitive Signature Scheme, [14,16]) A transitive signature scheme for directed trees is a tuple* $\mathtt{DTTS} = (\mathsf{TSKG}, \mathsf{TSign}, \mathsf{TSComp}, \mathsf{TSVf})$ *where:*

- $\mathsf{TSKG}(1^\kappa)$: *returns a pair of private/public keys* (tsk, tpk).
- $\mathsf{TSign}(tsk, a, b)$: *returns the signature* $\tau_{(a,b)}$ *of edge* (a, b).
- $\mathsf{TSComp}((a, b), \tau_{(a,b)}, (b, c), \tau_{(b,c)}, tpk)$*: returns a combined signature* $\tau_{(a,c)}$ *on edge* (a, c)*. Note that the secret key is not required.*
- $\mathsf{TSVf}((a, b), \tau, tpk)$: *returns* valid *if* τ *is a valid signature for the path* (a, b) *and* $\perp$ *otherwise.*

A transitive signature scheme is correct if both original signatures (those generated honestly with TSign) and combined signatures (those generated honestly with TSComp) do verify correctly with TSVf. Intuitively, a transitive signature scheme is secure if, for any PPT adversary it is infeasible to compute a signature for a path outside the transitive closure of $\mathcal{T}$.

Definition 2. *(Security of Transitive Signature Schemes, [14,16]) Let* $\mathtt{DTTS}$ *be a transitive signature scheme. Consider the following experiment. A PPT adversary* $\mathcal{A}$ *is given the public key tpk of the scheme.* $\mathcal{A}$ *may ask for a polynomial number of edge signatures to the oracle* $\mathcal{O}_{\mathsf{TSign}}(\cdot)$*. Finally* $\mathcal{A}$ *outputs* (a, b) *and* τ *where* a, b *are nodes in the tree* $\mathcal{T}$ *formed by the successive validly signed edges. The advantage of* $\mathcal{A}$ *is defined by:*

$$Adv^{tuf\text{-}cma}(\mathcal{A},\kappa) = Pr\left[\begin{array}{c}(a,b)\notin\mathcal{T}^*\wedge\\ \mathsf{TSVf}((a,b),\tau,tpk)=\mathsf{valid}\end{array}\right]$$

The scheme is said to be secure if we have $Adv^{tuf\text{-}cma}(\mathcal{A},\kappa) = \mathsf{neg}(\kappa)$ *for any PPT adversary* $\mathcal{A}$.

A trivial solution for *DTTS* can be implemented by simply concatenating standard edge signatures, in which case the size of a signature grows linearly with the length of the path and may reach $O(n\kappa)$ bits. Yi's solution [19] with $O(n\log(n\log n))$-bit signatures is clearly better than the trivial construction. Neven's *DTTS* scheme [16] manages to reduce signature sizes to $O(n\log n)$ bits. We remark that both solutions need to maintain the state of the tree to compute new edge signatures (as opposed to the initial definition [14]).

Our solution, like previous ones, is stateful. This state is shared by the *signer* and a *combiner*. The role of this new participant (the *combiner*) is to compute, *without help from the signer*, signatures for any edge in the transitive closure of the tree.

Bilinear maps. Our construction for *hashing with common-prefix proofs* requires the use of bilinear maps. Let $\mathbb{G},\mathbb{G}_T$, be cyclic groups of prime order p. We consider a map $e:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$ which is

- *bilinear*: $\forall a,b\in\mathbb{G}, x,y\in\mathbb{Z}_p : e(a^x,b^y)=e(a,b)^{xy}$.
- *non-degenerate*: let g be a generator of $\mathbb{G}$ then $e(g,g)$ also generates $\mathbb{G}_T$.
- *efficiently computable*: there exists a polynomial time algorithm BMGen with parameter 1^κ that outputs $(p,\hat{\mathbb{G}},\hat{\mathbb{G}_T},\hat{e},g)$ where $\hat{\mathbb{G}},\hat{\mathbb{G}_T}$ is the representation of the corresponding groups of size p (p being a prime number of κ bits), g is a generator of $\mathbb{G}$, and $\hat{e}$ is an efficient algorithm to compute the map. For the sake of simplicity, we will not distinguish between $\mathbb{G},\mathbb{G}_T,e$, and $\hat{\mathbb{G}},\hat{\mathbb{G}_T},\hat{e}$.

The security of our construction relies on the q-Billinear Diffie-Hellman Inversion (q-*BDHI*) assumption which was introduced by Boneh and Boyen [3].

Definition 3. *(q-BDHI assumption [3]) Let* $P=(p,\mathbb{G},\mathbb{G}_T,e,g)\leftarrow\mathsf{BMGen}(1^\kappa)$, $s\xleftarrow{R}\mathbb{Z}_p$, *and* $T=(g,g^s,g^{s^2},...,g^{s^q})$ *for* $q\in\mathbb{N}$. *The q-Bilinear Diffie-Hellman Inversion (q-BDHI) problem consists in computing* $e(g,g)^{\frac{1}{s}}$, *given* P *and* T. *We say the q-BDHI assumption holds if for any PPT adversary* $\mathcal{A}$ *we have:*

$$Adv^{q\text{-}BDHI}(\mathcal{A},\kappa,q) = \Pr\left[e(g,g)^{\frac{1}{s}}\leftarrow\mathcal{A}(1^\kappa,P,T)\right] = \mathsf{neg}(\kappa)$$

3 Collision-Resistant Hashing with Common-Prefix Proofs

Standard collision-resistant hash functions have the property of compressing possibly large inputs strings to small ones. Moreover, because of collision resistance, in practice hash functions are considered injective. This makes them

useful constructs to manipulate shorter strings without loosing much security. In that context, proving some relations or predicates on pre-images using only the corresponding hash values (and perhaps an additional short proof) is certainly very useful. For example, given two hash values $H(A), H(B)$, proving efficiently predicates like $|A - B| \geq 10$ or $A < B$ may help to simplify some protocols or make them more efficient.

With the above goal in mind, in this work we consider a predicate for strings called `CommonPrefix`: given $A, B \in \Sigma^m$, for some $m \in \mathbb{N}$, $\texttt{CommonPrefix}(A, B, i) = \mathit{true}$ if and only if A and B share a common prefix up to position i. More concretely, we seek collision-resistant hash function families $\mathcal{H}$ with the following property: given $H(A)$ and $H(B)$ where A and B share a common prefix until position i, it should be possible to produce a certificate π such that running a verification algorithm on inputs $H(A), H(B), i, \pi$ one can be convinced that $\texttt{CommonPrefix}(A, B, i) = \mathit{true}$. Any such scheme should be secure in the sense that if $\texttt{CommonPrefix}(A, B, i) = \mathit{false}$ then forging a proof π^* that makes the verification algorithm accept should be computationally infeasible. Clearly, there exist trivial instantiations of this primitive: just consider H a standard (collision-resistant) hash function and $\pi = (A, B)$. Of course, this is not really useful as the size of the certificate is proportional to the size of the longest string. Thus, interesting implementations should have short certificates. Additionally, we want hash function H to be efficiently updatable: given $H(A)$ one should be able to compute $H(A||\sigma)$ for any string σ, without knowing A (this concept is also known as *incremental hashing* [1]).

Given a `CommonPrefix` predicate we can now implement more interesting predicates over strings such as `Compare`, where $\texttt{Compare}(A, B) = \mathit{true}$ if and only if $A \prec B$ (where $\prec$ is the extended lexicographical order): If $A \prec B$ it follows that there exists a (possibly empty) common prefix C for A and B, such that (1) $D = C||\sigma$ is a prefix of A, (2) $C||\sigma'$ is a prefix of B, and (3) $\sigma < \sigma'$. In summary, once we know how to do short proofs for `CommonPrefix`, using incremental hashing we can compare any two strings by only their hash values.

Hashing with common-prefix proofs. Let $\kappa \in \{0,1\}^*$ be the security parameter, $PK \in \{0,1\}^\kappa$ some public key, and $n \in \mathbb{N}$ a bound on the size of the input[1] which is a polynomial in κ. We denote by $\mathcal{H} = \{\mathcal{H}_{PK,n,\kappa}\}$ a hash function family.

Definition 4. *(Hashing with common-prefix proofs - Syntax) A family $\mathcal{H}$ of hash functions with common-prefix proofs (HCPP) is a 4-tuple of algorithms* $(\mathsf{HGen}, \mathsf{HEval}, \mathsf{HProofGen}, \mathsf{HCheck})$ *where:*

- $\mathsf{HGen}(1^\kappa, n)$*: given a bound n on the length of the strings to hash, this probabilistic algorithm returns a public parameter PK. Value PK implicitly defines a hash function $H = H_{PK,n,\kappa} \in \mathcal{H}$ where $H : \{0,1\}^n \to \{0,1\}^\kappa$.*

[1] Here we use intentionally the same variable name n for the size of the input of the hash function as well as the number of nodes of the tree. Indeed, our full construction for trees of n nodes presented in section 4.2 requires hashing n-bit strings.

- $\mathsf{HEval}(M, PK)$*: given* $M \in \{0,1\}^n$*, this* deterministic *algorithm efficiently computes and returns the string* $H(M) \in \{0,1\}^\kappa$.
- $\mathsf{HProofGen}(A, B, i, PK)$*: given two messages* A, $B \in \{0,1\}^n$*, and an index* $1 \leq i \leq n$*, this deterministic algorithm computes a proof* $\pi \in \{0,1\}^\kappa$ *that will be used by the* HCheck *algorithm.*
- $\mathsf{HCheck}(H_A, H_B, \pi, i, PK)$*: a deterministic algorithm that, given* $H_A, H_B \in \{0,1\}^\kappa$*, two hash values, and a proof* $\pi \in \{0,1\}^\kappa$*, returns either* valid *or* $\perp$.

The scheme is said to be correct if for any strings A, B and $i \in \mathbb{N}$ such that $\mathtt{CommonPrefix}(A, B, i) = true$, and $\pi = \mathsf{HProofGen}(A, B, i, PK)$, we have that HCheck on inputs $(H(A), H(B), \pi, i, PK)$ returns valid.

The notion of security is also rather natural: for any PPT adversary $\mathcal{A}$ it should be difficult to compute two n-bit strings A, B, an index $i \in \{1, \ldots, n\}$, and a proof $\pi \in \{0,1\}^\kappa$ such that $\mathsf{HCheck}(H(A), H(B), \pi, i, PK)$ returns valid but $A[1..i] \neq B[1..i]$. Note that the adversary is required to output pre-images A and B to win, which guarantees that the hash values $H(A)$ and $H(B)$ have been correctly computed.

Definition 5. *(HCPP Security) Let* $\mathcal{H}$ *be a family of hash functions with common-prefix proofs and* $\mathcal{A}$ *a PPT adversary. The HCPP advantage of* $\mathcal{A}$ *is*

$$Adv_{\mathcal{H}}^{HCPP}(\mathcal{A}, \kappa, n) = Pr\left[\begin{array}{c} PK \leftarrow \mathsf{HGen}(1^\kappa, n); A, B, \pi, i \leftarrow \mathcal{A}(1^\kappa, n, PK) : \\ A[1..i] \neq B[1..i] \wedge H_A = H(A) \wedge H_B = H(B) \wedge \\ \mathsf{HCheck}(H_A, H_B, \pi, i, PK) = \mathsf{valid} \end{array}\right]$$

We say $\mathcal{H}$ *is a secure hash function family with common-prefix proofs (HCPP) if for every PPT* $\mathcal{A}$*, we have* $Adv_{\mathcal{H}}^{HCPP}(\mathcal{A}, \kappa, n) = \mathsf{neg}(\kappa)$.

The following proposition states that *hashing with common-prefix proofs* implies (standard) *collision resistance*. We omit the proof.

Proposition 1. *Let* $\mathcal{H}$ *be a family of hash functions with common-prefix proofs. Then* $\mathcal{H}$ *is a collision-resistant hash function family.*

The Construction. We assume that the description of the hash function H – i.e. the tuple $(g^s, g^{s^2}, ..., g^{s^n})$ of the *n-BDHI* problem – has been computed securely by a trusted third party or using multi-party computations techniques. The basic idea is to represent a binary string M by $H(M) \stackrel{\text{def}}{=} g^{M[1]s} \cdot g^{M[2]s^2} \cdots g^{M[n]s^n}$. Now if some message M' is equal to M up to position i then the value $\Delta = \frac{H(M)}{H(M')} = \prod_{j=i+1}^{n} g^{c_j s^j}$, where $c_j \in \{-1, 0, 1\}$, will be a product of powers of g^{s^j} for $1 \leq j \leq n$ where for all $j \leq i$ the exponents are equal to 0. Yet, the related value $\pi = \prod_{j=i+1}^{n} g^{c_j s^{j-(i+1)}}$ can easily be computed given M,M' and H. The intuition behind the proof is that as M and M' are equal up to position i then we can represent the difference between M and M' using only $n - i$ positions. Thus, verifying proof π simply consists in testing if using the bilinear map we can "shift forward" the exponents in the proof by i positions, to obtain Δ. More precisely, π will be a valid proof for $H(M), H(M')$ if and only if $e(\frac{H(M)}{H(M')}, g) = e(\pi, g^{s^{i+1}})$. Details follow.

Definition 6. *(Hashing with Common-Prefix Proofs - Scheme) Let* PH *be the scheme defined by the following algorithms:*

- $\mathsf{HGen}(1^\kappa, n)$*: Run* $\mathsf{BMGen}(1^\kappa)$ *to obtain* $P = (p, \mathbb{G}, \mathbb{G}_T, e, g)$*. Let* $s \xleftarrow{R} \mathbb{Z}_p$*, and* $T = (g, g^s, g^{s^2}, ..., g^{s^n})$*. Return* $PK = (P, T)$*.*
- $\mathsf{HEval}(M, PK)$*:* $M \in \{0,1\}^n$*. Compute* $H(M) = \prod_{j=1}^{n} g^{M[j]s^j}$*. Return* $H(M)$*.*
- $\mathsf{HProofGen}(A, B, i, PK)$*: Given* n*-bits strings* A, B*, let* C *be the array such that* $\forall j \in \{1, \ldots, n\} : C[j] = A[j] - B[j]$*. Return* $\pi = \prod_{j=i+1}^{n} g^{C[j]s^{j-(i+1)}}$*.*
- $\mathsf{HCheck}(H_A, H_B, \pi, i, PK)$*: Compute* $\Delta = \frac{H_A}{H_B}$*. If* $i = n$ *and* $\Delta = 1$ *return* valid*. If* $i < n$ *return* valid *if* $e(\Delta, g) = e(\pi, g^{s^{i+1}})$*, otherwise return* $\perp$*.*

Proposition 2. *Under the* n*-BDHI assumption the hash functions family defined by the scheme* PH *is HCPP secure.*

Proof. Given an adversary $\mathcal{A}$ that breaks the $HCPP$ security of PH, we construct an adversary $\mathcal{B}$ that breaks the n-$BDHI$ assumption as follows. Once $\mathcal{B}$ receives the parameters $(P, T = (g, g^s, g^{s^2}, \ldots, g^{s^n}))$ as input, it forwards them to $\mathcal{A}$. Eventually $\mathcal{A}$ will output values A, B, π, i such that $\mathsf{HCheck}(H(A), H(B), \pi, i, PK) = \mathsf{valid}$. Then, $\mathcal{B}$ computes the array C defined as $C[j] = A[j] - B[j] = c_j$ for $j \in \{1, \ldots, n\}$. Let k be the smallest index such that $c_k \neq 0$. Clearly $i - k > 0$ since $A[1..i] \neq B[1..i]$. From the validity of π, we have that $e(\Delta, g) = e(\pi, g^{s^{i+1}})$, and thus $\pi = \Delta^{\frac{1}{s^{i+1}}}$. Then:

$$\begin{aligned} E = e(\pi, g^{s^{i-k}}) &= e(\Delta^{\frac{1}{s^{i+1}}}, g^{s^{i-k}}) \\ &= \prod_{j=k}^{n} e(g,g)^{c_j s^{j-k-1}} \\ &= e(g,g)^{\frac{c_k}{s}} \prod_{j=k+1}^{n} e(g,g)^{c_j s^{j-k-1}} \\ &= e(g,g)^{\frac{c_k}{s}} D \end{aligned}$$

As all c_j are known, and $c_k = \pm 1$, $\mathcal{B}$ can compute $(\frac{E}{D})^{1/c_k} = e(g,g)^{\frac{1}{s}}$.

Additional properties. The $HCPP$ family $\mathcal{H}$ in our construction is homomorphic in the following sense: for any $H \in \mathcal{H}$, and any bit b, $H(M||b) = H(M) \cdot H(0^{|M|}||b)$. Moreover, since $H(0^{|M|}||b) = g^{bs^{|M|+1}}$ can be computed in constant time w.r.t $|M|$, our construction yields in fact an incremental hash function [1]. We call such combination of properties *incremental hashing with common-prefix proofs* and it is what we actually require in our most efficient construction. In terms of efficiency, both the computation of the hash function and the proof can be easily parallelizable as they involve only group *multiplications*. In particular, with $O(n)$ processors, we can compute a proof using only $O(\log n)$ (sequential) group multiplications. Also, note that handling strings of length $m > n$ can be done dynamically, *without having to recompute any proof*, by simply extending the public parameter $T = (g, g^s, g^{s^2}, \ldots, g^{s^n})$ say by invoking the distributed procedure (or calling the trusted generator) to compute $g^{s^{n+1}}, \ldots, g^{s^m}$.

4 Short Transitive Signatures for Directed Trees

Our construction for *DTTS* is based on the following idea: handling a growing tree can be reduced to maintaining two ordered sequences, one corresponding to the pre-order tree traversal and another to the post-order tree traversal in a depth-first search. This was first observed by Dietz [9].

Proposition 3. *([9]) Let $\mathcal{T}$ be a tree of n nodes and consider a depth-first traversal. Let* `Pre` *and* `Post` *be the strings formed by the nodes that are visited in pre-order and post-order respectively, then for any pair of nodes a, b in $\mathcal{T}$, b is descendant of a if and only if $\exists i, j : 0 < i < j$ and $\exists i', j' : 0 < j' < i'$ such that:*

$$(\texttt{Pre}[i] = a \wedge \texttt{Pre}[j] = b) \wedge (\texttt{Post}[i'] = a \wedge \texttt{Post}[j'] = b)$$

For example, consider the tree depicted in Fig. 1, last row, first column. For that tree, `Pre` $= acdbe$ and `Post` $= dcbea$. Since there is a path from c to d, c appears before d in `Pre` and d appears before c in `Post`. Also note that if there is no path from some node x to another node y then y may appear before x in `Pre` or x may appear before y in `Post`. See for example pairs $(c, b), (e, d)$ or (b, a).

The challenge in using this result is that the ordered sequences are dynamic – new elements can be inserted between any two existent elements. This problem is addressed by the so called *order data structure* [9,15]. Such a data structure allow us to compare any pair of labels and also insert a new label so that it may lie between two existing ones. A naive way - mentioned in [9] - to implement the proposed data structure would be to consider the interval $[0..2^n - 1]$ for the labels; to insert an element between X and Y one would use label $Z = \lfloor \frac{X+Y}{2} \rfloor$. This way we can always find the label for an element between any two others and the comparison algorithm consists in comparing the labels. Unfortunately, this solution does not suffice for our application since the string representation of a new label cannot be easily obtained from already computed labels, and the *signer* must sign labels of length n for each new edge. So our first improvement is a new order data structure with the following property: If X and Y are two consecutive labels,then every new computed label Z such that $X \prec Z \prec Y$ will share all bits except one with X or Y.

Before describing our construction we introduce the formal definition of order data structure [15]. Jumping ahead, we use this data structure to efficiently create and update labels in $\mathcal{U} = \{0, 1\}^*$, as well as to compare them using extended lexicographical order as the relation $\prec_{\mathcal{U}}$. The particular mapping between elements in lists `Pre` and `Post` to labels will depend on our construction.

Definition 7. *Let $(\mathcal{U}, \prec_{\mathcal{U}})$ be a totally ordered set of labels. An order data structure over $\mathcal{U}$ consists of three algorithms:*

- ODSetup() : *initializes the data structure.*
- ODInsert(X, Y) : *Let X and Y be two consecutive labels. Compute a new label Z such that $X \prec_{\mathcal{U}} Z \prec_{\mathcal{U}} Y$.*
- ODCompare(X, Y)*: returns true if and only if $X \prec_{\mathcal{U}} Y$.*

Our construction for order data structure uses binary tries [10] to handle labels. As mentioned before, in a trie, the label for a node is obtained by concatenating the labels on the edges in a path from the root to the node. Comparing two labels then reduces to comparing the labels as binary strings w.r.t. their extended lexicographical order.

Construction 1. *Let* `OrderDS` *be the order data structure over universe* $(\{0,1\}^*, \prec)$ *defined by the following operations:*

- $\mathsf{ODSetup}()$*: create a trie* $\mathcal{B}$ *with a single root node* r*. The label associated to the root node is* $R = \epsilon$*.*
- $\mathsf{ODInsert}(X, Y)$ *: If* $\mathcal{B}$ *has only one node with label* R *(root) then if* $X = R$ *(thus* Y *can be ignored) add a node node*(Z) *as the right child of* R*. Return* $Z = R||1$*. If* $Y = R$ *then add node*(Z) *as the left child of* R*. Return* $Z = R||0$*. If* $\mathcal{B}$ *has at least two nodes, then search node*(X) *and node*(Y)*, the nodes associated to labels* X *and* Y*, in the binary tree. If node*(Y) *belongs to the right sub-tree of node*(X) *then add node*(Z) *as the left child of node*(Y)*; Return* $Z = Y||0$*.*
  If node(X) *belongs to the left sub-tree of node*(Y) *then add node*(Z) *as the right child of node*(X)*. Return* $Z = X||1$*.*
- $\mathsf{ODCompare}(X, Y)$ *: Return true if and only if* $X \prec Y$*.*

In terms of efficiency, we observe that in the worst case, the longest path of a n-node tree is n, so labels are $O(n)$ bits. It is also easy to see that for any pair of consecutive labels X, Y the label Z returned by $\mathsf{ODInsert}(X, Y)$ is equal to $X||b$ or $Y||b$ where $b \in \{0,1\}$. Looking ahead, this property turns out to be crucial to obtain our most efficient construction as these strings will be *compressed* using the hash function with common-prefix proofs $\mathcal{H}$ introduced in the previous section. Moreover, the homomorphic property of $\mathcal{H}$ will allow us to compute $H(Z)$ from $H(X)$ or $H(Y)$ in only a constant number of cryptographic operations.

4.1 Basic Construction

Our first construction is based only on standard digital signatures, as Neven's construction. The scheme is as follows. Each time an edge (and thus a vertex) is inserted into the tree $\mathcal{T}$ two lists (one for pre-order `Pre` and another for post-order `Post`) are updated with the newly inserted element. We efficiently implement the latter by maintaining two order data structures $\mathtt{OrderDS}_{\mathtt{Pre}}$ and $\mathtt{OrderDS}_{\mathtt{Post}}$, one for each list. More precisely, each element $x \in \mathcal{T}$ is associated with a label $X_{\mathtt{Pre}}$ (resp. $X_{\mathtt{Post}}$) computed by the order data structure for pre-order (resp. post-order). (As a convention, the label associated to each element $x \in \mathcal{T}$ for list `Pre` is denoted by $X_{\mathtt{Pre}}$, using the same symbol but in capital letter and indexed by the list name. Same for `Post`.) We then use $\mathsf{ODCompare}$ to evaluate if x appears before some y in `Pre` (resp. `Post`), which simply verifies that $X_{\mathtt{Pre}} \prec Y_{\mathtt{Pre}}$ and $Y_{\mathtt{Post}} \prec X_{\mathtt{Post}}$.

Construction 2. *(DTTS from Standard Digital Signatures)*
Let $\mathtt{SSig} = (\mathsf{SKG}, \mathsf{SSig}, \mathsf{SVf})$ *be a standard digital signature scheme. The scheme* BasicDTTS *is as follows.*

- $\mathsf{TSKG}(1^\kappa)$: *Run* SKG *to generate a pair of keys* (sk, pk)*. Set* $tsk = sk$ *and* $tpk = pk$*. Initialize two order data structures* $\mathtt{OrderDS}_{\mathtt{Pre}}$ *and* $\mathtt{OrderDS}_{\mathtt{Post}}$ *to maintain the sequences for pre-order and post-order tree traversal respectively. Set* $\mathcal{T} = (V, E)$ *as the tree with a single root* r*. We define two tables* $\ell_{\mathtt{Pre}}[\cdot]$ *and* $\ell_{\mathtt{Post}}[\cdot]$ *that map nodes in* $\mathcal{T}$ *to their respective labels in* $\mathtt{OrderDS}_{\mathtt{Pre}}$ *and* $\mathtt{OrderDS}_{\mathtt{Post}}$ *respectively. That is, if* $x \in V$*,* $\ell_{\mathtt{Pre}}[x]$ *will return* $X_{\mathtt{Pre}}$ *and* $\ell_{\mathtt{Post}}[x]$ *will return* $X_{\mathtt{Post}}$*, the labels bound to* x *in* $\mathtt{OrderDS}_{\mathtt{Pre}}$ *and* $\mathtt{OrderDS}_{\mathtt{Post}}$ *respectively. We set* $\ell_{\mathtt{Pre}}[r] = \ell_{\mathtt{Post}}[r] = \epsilon$*. Return* (tsk, tpk)*.*
- $\mathsf{TSign}(tsk, a, b)$: *If both* $a, b \in V$ *or both* $a, b \notin V$ *or if the insertion does not preserve the tree structure of* $\mathcal{T}$*, return* $\perp$*. (Recall that tree* $\mathcal{T}$ *is initialized with a root node, so* a, b *should never be both not in* V*.)*
  Otherwise, let $z \in \{a, b\}$ *be the new vertex not yet in* V *and* $x \in \{a, b\} \setminus \{z\}$ *be the other one. Insert edge* (a, b) *in* $\mathcal{T}$*, and update* $\mathtt{OrderDS}_{\mathtt{Pre}}, \mathtt{OrderDS}_{\mathtt{Post}}$ *data structures to reflect the new pre-order and post-order tree traversal of* $\mathcal{T}$ *as follows. Let* y *be the element in* Pre *such that* z *lies (strictly) between* x *and* y*. Assume that* $x < z < y$ *(the other case is similar). Let* $X_{\mathtt{Pre}} = \ell_{\mathtt{Pre}}[x]$ *and* $Y_{\mathtt{Pre}} = \ell_{\mathtt{Pre}}[y]$*. Compute* $\mathtt{OrderDS}_{\mathtt{Pre}}.\mathsf{ODInsert}(X_{\mathtt{Pre}}, Y_{\mathtt{Pre}})$ *to obtain* $Z_{\mathtt{Pre}}$ *the label associated to* z *in* $\mathtt{OrderDS}_{\mathtt{Pre}}$*. Similarly obtain* $Z_{\mathtt{Post}}$ *the label associated to* z *in* $\mathtt{OrderDS}_{\mathtt{Post}}$*. Set* $\ell_{\mathtt{Pre}}[z] = Z_{\mathtt{Pre}}$ *and* $\ell_{\mathtt{Post}}[z] = Z_{\mathtt{Post}}$*. Then, compute* $M_z = z||Z_{\mathtt{Pre}}||Z_{\mathtt{Post}}$ *and sign it to obtain* $\sigma_z = \mathsf{SSig}(tsk, M_z)$*. Now, obtain* $X_{\mathtt{Post}} = \ell_{\mathtt{Post}}[x]$*, and (re)compute signature* $\sigma_x = \mathsf{SSig}(tsk, M_x)$ *on* $M_x = x||X_{\mathtt{Pre}}||X_{\mathtt{Post}}$*. Return* $\tau_{(x,z)} = (M_x, \sigma_x, M_z, \sigma_z)$*.*
- $\mathsf{TSComp}((a, b), \tau_{(a,b)}, (b, c), \tau_{(b,c)}, tpk)$*: Parse* $\tau_{(a,b)}$ *as* $(M_a, \sigma_a, M_b, \sigma_b)$ *and* $\tau_{(b,c)}$ *as* $(M_b, \sigma_b, M_c, \sigma_c)$*. Return* $\tau_{(a,c)} = (M_a, \sigma_a, M_c, \sigma_c)$*.*
- $\mathsf{TSVf}((a, b), \tau, tpk)$: *Parse* τ *as* $(M_a, \sigma_a, M_b, \sigma_b)$*. If* M_a *or* M_b *are not of the form* $(a, A_{\mathtt{Pre}}, A_{\mathtt{Post}})$ *or* $(b, B_{\mathtt{Pre}}, B_{\mathtt{Post}})$ *respectively, or if any signatures is invalid, then return* $\perp$*. Otherwise, verify that* a *appears before (resp. after)* b *in* Pre *(resp.* Post*) by checking that* $\mathsf{ODCompare}(A_{\mathtt{Pre}}, B_{\mathtt{Pre}}) = true$ *and* $\mathsf{ODCompare}(B_{\mathtt{Post}}, A_{\mathtt{Post}}) = true$*. If verification succeeds return* valid *else return* $\perp$*.*

Correctness and Security. We require that correct signatures, those honestly computed by the *signer* as well as those combined by anyone from two existent signatures, verify correctly, meaning the verification algorithm on them returns valid. To see this holds, it suffices to observe first, that the signing operation preserves the tree structure of the graph, and second, that $\mathsf{ODCompare}(X, Y)$ is true, if and only if $x = \ell^{-1}_{\mathtt{Pre}}[X]$ (resp. $y = \ell^{-1}_{\mathtt{Post}}[Y]$) appears before $y = \ell^{-1}_{\mathtt{Pre}}[Y]$ (resp. $x = \ell^{-1}_{\mathtt{Post}}[X]$) in Pre (resp. Post) which allows correct verification by Dietz's condition (namely, proposition 3). Our scheme is secure if the underlying standard signature scheme is secure.

Theorem 1. *If* SSig *is a signature scheme existentially unforgeable under chosen message attack then* BasicDTTS *a is secure transitive signature scheme for directed trees where edge signatures are* $O(n + \kappa)$ *bits long.*

Step	Tree $\mathcal{T}$	Pre/Post order traversal	$\texttt{OrderDS}_{\texttt{Pre}}$	$\texttt{OrderDS}_{\texttt{Post}}$	Labels
0	ϵ	$\texttt{Pre} = \epsilon$ $\texttt{Post} = \epsilon$	r $\epsilon\ \epsilon$	r $\epsilon\ \epsilon$	$\ell_{\texttt{Pre}}[r] = \epsilon$ $\ell_{\texttt{Post}}[r] = \epsilon$
1	a $\epsilon\ \epsilon$	$\texttt{Pre} = a$ $\texttt{Post} = a$	a $\epsilon\ \epsilon$	a $\epsilon\ \epsilon$	$A_{\texttt{Pre}} = \epsilon$ $A_{\texttt{Post}} = \epsilon$
2	a ϵ b	$\texttt{Pre} = ab$ $\texttt{Post} = ba$	a ϵ b	a b ϵ	$B_{\texttt{Pre}} = 1$ $B_{\texttt{Post}} = 0$
3	a c b	$\texttt{Pre} = acb$ $\texttt{Post} = cba$	a ϵ b c ϵ	a b ϵ c ϵ	$C_{\texttt{Pre}} = 10$ $C_{\texttt{Post}} = 00$
4	a c b d	$\texttt{Pre} = acdb$ $\texttt{Post} = dcba$	a ϵ b c ϵ ϵ d	a b ϵ c ϵ d ϵ	$D_{\texttt{Pre}} = 101$ $D_{\texttt{Post}} = 000$
5	a c b e d	$\texttt{Pre} = acdbe$ $\texttt{Post} = dcbea$	a ϵ b c e ϵ d	a b ϵ c e d ϵ	$E_{\texttt{Pre}} = 11$ $E_{\texttt{Post}} = 01$

Fig. 1. Example of order data structures after several insertions in a directed tree $\mathcal{T}$
Step 0: The tree $\mathcal{T}$ to authenticate has no nodes. The sequences `Pre` and `Post` are empty as well. The order data structure $\texttt{OrderDS}_{\texttt{Pre}}$ and $\texttt{OrderDS}_{\texttt{Post}}$ contain a single root node r. Each edge is marked implicitly by 0 (left child) and 1 (right child).
Step 1: The first node a of $\mathcal{T}$ is created. The pre/post-order lists contain only a. The order data structures are updated by setting node a to be equal to r. In particular we have that labels $A_{\texttt{Pre}} = A_{\texttt{Post}} = \epsilon$.
Step 2: A child b is added to a. Now the pre-order sequence `Pre` is equal to ab and the post-order sequence is ba. As b comes after a in `Pre` we have that b is the right child of a in $\texttt{OrderDS}_{\texttt{Pre}}$. Similarly b is the left child of a in $\texttt{OrderDS}_{\texttt{Post}}$ as it comes before a in `Post`.
Step 3,4 and 5: We follow the same procedure and obtain for each node x its order labels $X_{\texttt{Pre}}$ and $X_{\texttt{Post}}$ respectively.

Comparing Two Node Labels: In step 5 we can check easily using the order labels that for example d is a descendant of a. Indeed we have $A_{\texttt{Pre}} = \epsilon$ and $D_{\texttt{Pre}} = 101$ which means $A_{\texttt{Pre}} \prec D_{\texttt{Pre}}$. Also we can check that $D_{\texttt{Post}} = 000 \prec A_{\texttt{Post}} = \epsilon$. We can also observe that there is no path from b to c for example as $C_{\texttt{Pre}} = 10 \prec B_{\texttt{Pre}} = 1$ and also $C_{\texttt{Post}} = 00 \prec B_{\texttt{Post}} = 0$.

4.2 Full Construction

We extend our basic construction as follows. Instead of comparing strings directly, we compare them by proving that labels from certain nodes contain the labels of other nodes as prefixes. Such proofs are done using scheme `PH`. As before, we use alphabet $\Sigma = \{0, \$, 1\}$ where $0 < \$ < 1$. That is, in order to prove that two labels X, Y are such that $X \prec Y$ using their hashes $H(X), H(Y)$

instead of the strings, the *combiner* must compute: (1) the longest common prefix C for X and Y, (2) a proof that C is a prefix of X up to position $i = |C|$, (3) a proof that C is a prefix of Y up to position $i = |C|$, (4) a proof that $C||c_1$ is a prefix of $X||\$$ up to position $i+1$ for some $c_1 \in \Sigma$, and (5) a proof that $C||c_2$ is a prefix of $Y||\$$ up to position $i+1$, for some $c_2 \in \Sigma$. Then, verifying that $X \prec Y$ reduces to the checking of the proofs and verifying that $c_1 < c_2$.

Construction 3. *(DTTS from HCPP)*
Let $\mathtt{PH} = (\mathsf{HGen}, \mathsf{HEval}, \mathsf{HProofGen}, \mathsf{HCheck})$ *be a family of hash functions with common-prefix proofs. The scheme* PHDTTS *consists of following algorithms:*

- $\mathsf{TSKG}(1^\kappa)$: *Do as in* BasicDTTS *and also generate the public parameters (ie. PK) for the* PH *scheme. Return* (tsk, tpk, PK).
- $\mathsf{TSign}(tsk, a, b)$: *Do as in* BasicDTTS *except that the message* M_z *to sign is now* $M_z = z||H_{Z_{\mathtt{Pre}}}||H_{Z_{\mathtt{Post}}}$, *where* $H_{Z_{\mathtt{Pre}}} = H(Z_{\mathtt{Pre}})$ *and* $H_{Z_{\mathtt{Post}}} = H(Z_{\mathtt{Post}})$. *Store* $H_{Z_{\mathtt{Pre}}}$ *and* $H_{Z_{\mathtt{Post}}}$ *on node* z *in the tree* $\mathcal{T}$. *Notice that* $H_{Z_{\mathtt{Pre}}}, H_{Z_{\mathtt{Post}}}$ *can be computed incrementally due to* H*'s homomorphism.*
- $\mathsf{TSComp}((a,b), \tau_{(a,b)}, (b,c), \tau_{(b,c)}, tpk)$: *Parse* $\tau_{(a,b)}$ *as* $(M_a, \sigma_a, M_b, \sigma_b)$ *and* $\tau_{(b,c)}$ *as* $(M_b, \sigma_b, M_c, \sigma_c)$. *If* σ_a, σ_b *or* σ_c *is invalid, then reject. Parse* $M_a = (a, H_{A_{\mathtt{Pre}}}, H_{A_{\mathtt{Post}}})$ *and* $M_c = (c, H_{C_{\mathtt{Pre}}}, H_{C_{\mathtt{Post}}})$. *Compute proof* $\pi_{\mathtt{Pre}}$ *as follows. First, save* $(H_{A_{\mathtt{Pre}}}, H_{A_{\mathtt{Post}}}, \sigma_a)$ *on node* a *in* $\mathcal{T}$. *Do the same with the information for node* b. *Recover* $A_{\mathtt{Pre}}$ *and* $C_{\mathtt{Pre}}$ *as the labels associated to* a *and* c *respectively from* $\mathtt{OrderDS}_{\mathtt{Pre}}$. *Let* $D_{\mathtt{Pre}}$ *be the longest common prefix between* $A_{\mathtt{Pre}}$ *and* $C_{\mathtt{Pre}}$. *Note that* $H(D_{\mathtt{Pre}})$ *has already been computed by the* signer *and thus* $M_d = d\ ||H(D_{\mathtt{Pre}})||H(D_{\mathtt{Post}})$, *signature* σ_d *on* M_d *are available to the* combiner. *Let* $t = |D_{\mathtt{Pre}}|$. *Compute the following values:*
  - $\pi_1 = \mathsf{HProofGen}(D_{\mathtt{Pre}}, A_{\mathtt{Pre}}, t, PK)$, $\pi_2 = \mathsf{HProofGen}(D_{\mathtt{Pre}}, C_{\mathtt{Pre}}, t, PK)$
  - $\pi_3 = \mathsf{HProofGen}(D_{\mathtt{Pre}}||d_1, A_{\mathtt{Pre}}||\$, t+1, PK)$, *and*
  - $\pi_4 = \mathsf{HProofGen}(D_{\mathtt{Pre}}||d_2, C_{\mathtt{Pre}}||\$, t+1, PK)$

  where $(d_1, d_2) \in \{(0,\$), (0,1), (\$,1)\}$. *Set* $\pi_5 =$ $(M_a, \sigma_a, M_c, \sigma_c, M_d, \sigma_d, t, d_1, d_2)$ *and* $\pi_{\mathtt{Pre}} = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$. *Compute similarly* $\pi_{\mathtt{Post}}$ *and return* $\tau_{(a,d)} = (\pi_{\mathtt{Pre}}, \pi_{\mathtt{Post}})$.
- $\mathsf{TSVf}((a,c), \tau, tpk, PK)$: *If* τ *is of the form* $(M_a, \sigma_a, M_c, \sigma_c)$, *verify that* M_a *is a* 3*-tuple that starts with* a *and* M_c *with* c, *and return* $\perp$ *if one of the signature* σ_a *or* σ_c *is invalid. Otherwise, extract* $\pi_{\mathtt{Pre}}$ *from* $\tau = (\pi_{\mathtt{Pre}}, \pi_{\mathtt{Post}})$. *Parse* $\pi_{\mathtt{Pre}}$ *as* $(\pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$. *Parse* π_5 *as* $(M_a, \sigma_a, M_c, \sigma_c,\ M_d, \sigma_d, t, d_1, d_2)$ *where* $M_x = x||X_{\mathtt{Pre}}||X_{\mathtt{Post}}$ *for* $x \in \{a, c, d\}$. *Check that all (standard) signatures are valid under public key tpk. Check that* $d_1, d_2 \in \Sigma$ *and* $d_1 < d_2$. *Verify proofs* $\pi_1, \pi_2, \pi_3, \pi_4$ *using* HCheck*:*
  - $\mathsf{HCheck}(H_{D_{\mathtt{Pre}}}, H_{A_{\mathtt{Pre}}}, \pi_1, t, PK)$, *and* $\mathsf{HCheck}(H_{D_{\mathtt{Pre}}}, H_{C_{\mathtt{Pre}}}, \pi_2, t, PK)$,
  - $\mathsf{HCheck}(H_{D_{\mathtt{Pre}}} \cdot H(0^t||d_1), H_{A_{\mathtt{Pre}}} \cdot H(0^t||\$), \pi_3, t+1, PK)$, *and*
  - $\mathsf{HCheck}(H_{D_{\mathtt{Pre}}} \cdot H(0^t||d_2), H_{C_{\mathtt{Pre}}} \cdot H(0^t||\$), \pi_4, t+1, PK)$

  Perform the similar verifications relative to $\mathtt{OrderDS}_{\mathtt{Post}}$. *If all these verifications pass return* valid *otherwise return* $\perp$.

This new construction combines the basic one with hashing with common-prefix proofs so we can shrink the size of an edge signature to $O(\kappa)$ bits. Furthermore, using a tradeoff technique for our hashing family we obtain the following result[2].

Theorem 2. *Let $\lambda \geq 1$. If* `SSig` *is a signature scheme existentially unforgeable under chosen message attack and $\mathcal{H}$ is a family of secure hash functions with common-prefix proofs, then* `PHDTTS` *with tradeoff is a secure DTTS scheme. Moreover, (a) an edge signature is $O(\lambda\kappa)$ bits long can verified in $O(\lambda)$ cryptographic operations, (b) the* signer *has to perform $O(\lambda)$ cryptographic operations to sign an edge, and (c) computing the signature for any edge (in the transitive closure of the tree) takes $O(\lambda n^{1/\lambda})$ cryptographic operations for the* combiner.

5 Conclusion

In this work we propose a transitive signature scheme for directed trees which achieves better worst-case complexity than previously known constructions, and enables a practical trade off between the time to combine a signature, $O(\lambda n^{1/\lambda})$, and the time to verify it, $O(\lambda)$. However, the problem of building *short* and *stateless* transitive signatures for directed trees remains open. Moreover, in order to achieve the mentioned trade off, we introduce a new primitive *hash functions with common-prefix proofs*. We believe it may find other useful applications.

Acknowledgements. The authors would like to thank the anonymous referees for their valuable comments and suggestions to improve the quality of this paper.

References

1. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental Cryptography: The Case of Hashing and Signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
2. Bellare, M., Neven, G.: Transitive Signatures: New Schemes and Proofs. IEEE Transactions on Information Theory 51(6), 2133–2151 (2005)
3. 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)
4. Boneh, D., Freeman, D.: Homomorphic Signatures for Polynomial Functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
5. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a Linear Subspace: Signature Schemes for Network Coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)

[2] Due to space constrainsts, the description of the trade off technique for the hash function, its use in our signature scheme, and the security proofs are in the full version of this paper [6].

6. Camacho, P., Hevia, A.: Short Transitive Signatures for Directed Trees. Full version of this paper (2011), http://eprint.iacr.org/2011/438
7. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
8. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
9. Dietz, P.F.: Maintaining order in a linked list. In: STOC, pp. 122–127. ACM Press (1982)
10. Fredkin, E.: Trie memory. Communications of the ACM 3(9), 490–499 (1960)
11. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281 (1988)
12. Hohenberger, S.: The Cryptographic Impact of Groups with Infeasible Inversion. S.M. Thesis, MIT (May 2003)
13. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
14. Micali, S., Rivest, R.: Transitive Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 236–243. Springer, Heidelberg (2002)
15. Bender, M.A., Cole, R., Demaine, E.D., Farach-Colton, M., Zito, J.: Two simplified algorithms for maintaining order in a list. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 152–164. Springer, Heidelberg (2002)
16. Neven, G.: A simple transitive signature scheme for directed trees. Theoretical Computer Science 396(1-3), 277–282 (2008)
17. Shahandashti, S.F., Salmasizadeh, M., Mohajeri, J.: A Provably Secure Short Transitive Signature Scheme from Bilinear Group Pairs. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 60–76. Springer, Heidelberg (2005)
18. Xu, J.: On Directed Transitive Signature (2009), http://eprint.iacr.org/2009/209
19. Yi, X.: Directed Transitive Signature Scheme. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 129–144. Springer, Heidelberg (2006)

Short Attribute-Based Signatures for Threshold Predicates

Javier Herranz[1], Fabien Laguillaumie[2], Benoît Libert[3], and Carla Ràfols[4]

[1] Universitat Politècnica de Catalunya, Dept. Matemàtica Aplicada IV, Spain
[2] Université de Caen Basse - Normandie and CNRS/ENSL/INRIA/UCBL LIP, Lyon, France
[3] Université Catholique de Louvain, ICTEAM Institute – Crypto Group, Belgium
[4] Universitat Rovira i Virgili, UNESCO Chair in Data Privacy, Tarragona, Catalonia

Abstract. Attribute-based cryptography is a natural solution for fine-grained access control with respect to security policies. In the case of attribute-based signatures (ABS), users obtain from an authority their secret keys as a function of the attributes they hold, with which they can later sign messages for any predicate satisfied by their attributes. A verifier will be convinced of the fact that the signer's attributes satisfy the signing predicate while remaining completely ignorant of the identity of the signer. In many scenarios where authentication and anonymity are required, like distributed access control mechanisms in ad hoc networks, the bandwidth is a crucial and sensitive concern. The signatures' size of all previous ABS schemes grows linearly in the number of attributes involved in the signing predicate. We propose the first two attribute-based signature schemes with constant size signatures. Their security is proven in the selective-predicate and adaptive-message setting, in the standard model, under chosen message attacks, with respect to some algorithmic assumptions related to bilinear groups. The described schemes are for the case of threshold predicates, but they can be extended to admit some other (more expressive) kinds of monotone predicates.

1 Introduction

Attribute-based cryptography offers a real alternative to public-key cryptography when the systems to be protected also require anonymity among users following a security policy. In this setting, users obtain their secret keys from an authority as a function of their attributes. The operation involving the secret key proves somehow that the user holds a certain subset of attributes, without leaking information on his identity or on his total set of attributes.

One of the major issues in attribute-based cryptography is to save bandwidth, and in particular to get ciphertexts or signatures of constant size, *i.e.,* not depending on the number of involved attributes. Other important issues are the construction of systems achieving security in the strongest possible model and being as expressive as possible, *i.e.,* admitting a wide variety of policies. The goal of this work is to address the first question in the context of signature design.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 51–67, 2012.

Attribute-based cryptography first appeared in [15] with an attribute-based encryption scheme, as an extension of fuzzy identity-based cryptosystems [29]. Since then, the notion of attribute-based encryption (ABE for short, conjugated into *key policy* or *ciphertext policy*) has received a lot of attention (see, e.g., [2,17,20]), notably with attempts to compress ciphertexts (see [13,17,1]).

Attribute-based signatures (ABS) have been explicitly introduced more recently in [24] (see also [30,21,22]), although the idea was implicitly considered before (for instance, in [10]). They are related to the notion of (threshold) ring signatures [28,9] or mesh signatures [8], but offer much more flexibility and versatility to design secure complex systems, since the signatures are linked not to the users themselves, but to their attributes. As a consequence, these signatures have a wide range of applications, like private access control, anonymous credentials, trust negotiations, distributed access control mechanisms for ad hoc networks or attribute-based messaging (see [24] for detailed descriptions of applications). In terms of security, ABS must first satisfy unforgeability, which guarantees that a signature cannot be computed by a user who does not have the right attributes, even if he colludes with other users by pooling together their secret keys. The other security requirement is the privacy of user's attributes, in the sense that a signature should not leak any information about the actual attributes that have been employed to produce it.

Related work. The schemes proposed by Maji, Prabhakaran, Rosulek in [24] support very expressive signing predicates, but their most practical one is only proven secure in the generic group model. The scheme of [27] is claimed to be "almost optimally efficient", although its signatures' length grows linearly in the size of the span program (which is greater than the number of involved attributes in the signing predicate). Our result shows that specific families of predicates (e.g., threshold predicates) allow for more compact signatures. Other instantiations in [24] are secure in the standard model, but are substantially less inefficient (*i.e.,* signatures consist of a linear number of group elements in the security parameter) as they use Groth-Sahai proofs for relations between the bits of elements in the group. In the standard model, Okamoto and Takashima designed [27] a *fully* secure ABS which supports general non-monotone predicates. The scheme is built upon dual pairing vector spaces [26] and uses proof techniques from functional encryption [20]. Escala, Herranz and Morillo also proposed in [14] a fully secure ABS in the standard model, with the additional property of revocability, meaning that a third party can extract the identity of a signer in case of dispute (thanks to a secret that can be computed by the master entity). As it turns out, *none* of the previous schemes achieves constant-size signatures.

Our contribution. This paper describes the first two threshold ABS schemes featuring constant-size signatures and proves them secure in the selective-predicate setting (*i.e.,* as opposed to the *full* security setting) in the standard model. We hope our results will inspire ideas leading to the design of fully secure ABS schemes with constant-size signatures and supporting more expressive predicates than in this paper. The new schemes are built (non-generically) on two different

constant-size attribute-based encryption schemes. In both schemes, n denotes the maximum size of the admitted signing predicates.

- Our first scheme supports (weighted) threshold predicates for small[1] universes of attributes. Its design is inspired by the constant-size ciphertext-policy ABE scheme from [17] by Herranz, Laguillaumie and Ràfols, in the sense that the signer implicitly proves his ability to decrypt a ciphertext by using the Groth-Sahai proof systems [16], and by binding the signed message (and the corresponding predicate) to the signature using a technique suggested by Malkin, Teranishi, Vahlis and Yung [23]. The signature consists of 15 group elements, and the secret key of a user holding a set Ω of attributes has $|\Omega|+n$ elements. Our scheme is selective-predicate and adaptive-message unforgeable under chosen message attacks if the augmented multi-sequence of exponents computational Diffie-Hellman assumption [17] and the Decision Linear assumption [5] hold. The privacy of the attributes used to sign is proved in the computational sense under the Decision Linear assumption [5].
- The second scheme supports threshold predicates (as well as compartmented and hierarchical predicates) for *large* universes of attributes, which can be obtained by hashing arbitrary strings. It is built upon a key-policy ABE scheme proposed by Attrapadung, Libert and de Panafieu [1] and has signatures consisting of *only* 3 group elements. The secret keys are longer than in the first scheme, as they include $(2n+2)\times(|\Omega|+n)$ group elements. On the other hand, its selective-predicate and adaptive-message unforgeability relies on the more classical n-Diffie-Hellman exponent assumption. Moreover, the scheme protects the privacy of the involved attributes unconditionally.

Organization. Section 2 gives the algorithmic setting and defines the syntax and the security properties of attribute-based signatures. In Sections 3 and 4 we describe our two constructions for threshold predicates. Section 5 discusses extensions of both schemes to more general predicates.

2 Background

We will treat a vector as a column vector. For any $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \mathbb{Z}_p^n$, and any element g of a group $\mathbb{G}$, $g^{\boldsymbol{\alpha}}$ stands for $(g^{\alpha_1}, \ldots, g^{\alpha_n})^\top \in \mathbb{G}^n$. The inner product of $\boldsymbol{a}, \boldsymbol{z} \in \mathbb{Z}_p^n$ is denoted as $\langle \boldsymbol{a}, \boldsymbol{z} \rangle = \boldsymbol{a}^\top \boldsymbol{z}$. Given $g^{\boldsymbol{a}}$ and $\boldsymbol{z}$, $(g^{\boldsymbol{a}})^{\boldsymbol{z}} := g^{\langle \boldsymbol{a}, \boldsymbol{z} \rangle}$ is computable without knowing $\boldsymbol{a}$. For equal-dimension vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ of exponents or group elements, $\boldsymbol{A} \cdot \boldsymbol{B}$ stands for their component-wise product. We denote by I_n the identity matrix of size n. For any set U, we define $2^U = \{S \mid S \subseteq U\}$. Given a set $S \subset \mathbb{Z}_p$, and some $i \in S$, the i-th Lagrange basis polynomial is $\Delta_i^S(X) = \prod_{j \in S \setminus \{i\}} (X-j)/(i-j)$.

[1] *i.e.* polynomial in the security parameter, which is sufficient for many applications.

2.1 Complexity Assumptions

Our two schemes work in the setting of bilinear groups. That is, we use a pair of multiplicative groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p with an efficiently computable mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ s.t. $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$, $a, b \in \mathbb{Z}$ and $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

The security of our first scheme is partially based on the hardness of the computational version of a problem appeared in [17] under the name of *augmented multi-sequence of exponents decisional Diffie-Hellman problem*. Its decisional version was proven to be hard in generic groups.

Definition 1 ($(\tilde{\ell}, \tilde{m}, \tilde{t})$-aMSE-CDH - [17]). *The $(\tilde{\ell}, \tilde{m}, \tilde{t})$-augmented multi-sequence of exponents computational Diffie-Hellman ($(\tilde{\ell}, \tilde{m}, \tilde{t})$-aMSE-CDH) problem related to the group pair $(\mathbb{G}, \mathbb{G}_T)$ is to compute $T = e(g_0, h_0)^{\kappa \cdot f(\gamma)}$ when $\kappa, \alpha, \gamma, \omega$ are unknown random elements of $\mathbb{Z}_p$ and g_0 and h_0 are generators of $\mathbb{G}$ on input the vector $\boldsymbol{x}_{\tilde{\ell}+\tilde{m}} = (x_1, \ldots, x_{\tilde{\ell}+\tilde{m}})^\top$, whose components are pairwise distinct elements of $\mathbb{Z}_p$, and the values*

$$\begin{cases} g_0, g_0^{\gamma}, \ldots, g_0^{\gamma^{\tilde{\ell}+\tilde{t}-2}}, & g_0^{\kappa \cdot \gamma \cdot f(\gamma)}, \quad (1.1) \\ g_0^{\omega\gamma}, \ldots, g_0^{\omega\gamma^{\tilde{\ell}+\tilde{t}-2}}, & \quad (1.2) \\ g_0^{\alpha}, g_0^{\alpha\gamma}, \ldots, g_0^{\alpha\gamma^{\tilde{\ell}+\tilde{t}}}, & \quad (1.3) \\ h_0, h_0^{\gamma}, \ldots, h_0^{\gamma^{\tilde{m}-2}}, & h_0^{\kappa \cdot g(\gamma)} \quad (1.4) \\ h_0^{\omega}, h_0^{\omega\gamma}, \ldots, h_0^{\omega\gamma^{\tilde{m}-1}}, & \quad (1.5) \\ h_0^{\alpha}, h_0^{\alpha\gamma}, \ldots, h_0^{\alpha\gamma^{2(\tilde{m}-\tilde{t})+3}} & \quad (1.6), \end{cases}$$

where $f(X) = \prod_{i=1}^{\tilde{\ell}} (X + x_i)$ and $g(X) = \prod_{i=\tilde{\ell}+1}^{\tilde{\ell}+\tilde{m}} (X + x_i)$.

The security analysis of our first scheme also relies on the Decision Linear assumption.

Definition 2 ([5]). *In a group $\mathbb{G}$ of order p, the* Decision Linear Problem *(DLIN) is to distinguish the distributions $(g, g^a, g^b, g^{a \cdot \delta_1}, g^{b \cdot \delta_2}, g^{\delta_1 + \delta_2})$ and $(g, g^a, g^b, g^{a \cdot \delta_1}, g^{b \cdot \delta_2}, g^{\delta_3})$, with $a, b, \delta_1, \delta_2, \delta_3 \stackrel{R}{\leftarrow} \mathbb{Z}_p$.*

This problem is to decide if vectors $\boldsymbol{g_1} = (g^a, 1, g)^\top$, $\boldsymbol{g_2} = (1, g^b, g)^\top$ and $\boldsymbol{g_3} = (g^{a\delta_1}, g^{b\delta_2}, g^{\delta_3})^\top$ are linearly dependent in the $\mathbb{Z}_p$-module $\mathbb{G}^3$ formed by entry-wise multiplication.

The security of our second scheme is based on a non-interactive and falsifiable [25] assumption, the hardness of n-Diffie-Hellman Exponent problem, proven to hold in generic groups in [4].

Definition 3 ([6]). *In a group $\mathbb{G}$ of prime order p, the n-*Diffie-Hellman Exponent *(n-DHE) problem is, given a tuple $(g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^n}, g^{\gamma^{n+2}}, \ldots, g^{\gamma^{2n}})$ where $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p$, $g \stackrel{R}{\leftarrow} \mathbb{G}$, to compute $g^{\gamma^{n+1}}$.*

2.2 Groth-Sahai Proof Systems

To simplify the description, our first scheme uses Groth-Sahai proofs based on the DLIN assumption and symmetric pairings, although instantiations based on the symmetric external Diffie-Hellman assumption are also possible. In the DLIN setting, the Groth-Sahai proof systems [16] use a common reference string comprising vectors $\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g_3} \in \mathbb{G}^3$, where $\boldsymbol{g_1} = (g_1, 1, g)^\top$, $\boldsymbol{g_2} = (1, g_2, g)^\top$ for some $g_1, g_2, g \in \mathbb{G}$. To commit to $X \in \mathbb{G}$, one sets $\boldsymbol{C} = (1, 1, X)^\top \cdot \boldsymbol{g_1}^r \cdot \boldsymbol{g_2}^s \cdot \boldsymbol{g_3}^t$ with $r, s, t \xleftarrow{R} \mathbb{Z}_p$. In the soundness setting (*i.e.*, when proofs should be perfectly sound), $\boldsymbol{g_3}$ is set as $\boldsymbol{g_3} = \boldsymbol{g_1}^{\xi_1} \cdot \boldsymbol{g_2}^{\xi_2}$ with $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$. Commitments $\boldsymbol{C} = (g_1^{r+\xi_1 t}, g_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})^\top$ are then Boneh-Boyen-Shacham (BBS) ciphertexts [5] that can be decrypted using $a = \log_g(g_1)$, $b = \log_g(g_2)$.

In contrast, defining $\boldsymbol{g_3} = \boldsymbol{g_1}^{\xi_1} \cdot \boldsymbol{g_2}^{\xi_2} \cdot (1, 1, g^{-1})^\top$ gives linearly independent $\{\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g_3}\}$ and $\boldsymbol{C}$ is a perfectly hiding commitment. Moreover, proofs are perfectly witness indistinguishable (WI) in that two proofs generated using any two distinct witnesses are perfectly indistinguishable. Under the DLIN assumption, the WI and the soundness setting are computationally indistinguishable.

To prove that committed group elements satisfy certain relations, the Groth-Sahai techniques require one commitment per variable and one proof element (made of a constant number of group elements) per relation. Such proofs are available for pairing-product relations, which are of the type

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^{n} \cdot \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T, \tag{1}$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbb{G}$, $a_{ij} \in \mathbb{Z}_p$, for $i, j \in \{1, \ldots, n\}$.

At some additional cost (typically, auxiliary variables have to be introduced), pairing-product equations admit non-interactive zero-knowledge (NIZK) proofs (this is the case when the target element t_T has the special form $t_T = \prod_{i=1}^{t} e(S_i, T_i)$, for constants $\{(S_i, T_i)\}_{i=1}^{t}$ and some $t \in \mathbb{N}$): on a simulated common reference string (CRS), prepared for the WI setting, a trapdoor makes it possible to simulate proofs without knowing the witnesses. Linear pairing product equations (where $a_{ij} = 0$ for all i, j in (1)) consist of only 3 group elements and we only need linear equations here.

2.3 Syntax of Threshold Attribute-Based Signatures

We describe the syntax and security model of attribute-based signatures with respect to *threshold* signing predicates $\Gamma = (t, S)$, but the algorithms and security model for more general signing predicates can be described in a very similar way. In the threshold case, every message Msg is signed for a subset S of the universe of attributes and a threshold t such that $1 \leq t \leq |S|$ of the sender's choice.

An *attribute-based signature* ABS = (ABS.TSetup, ABS.MSetup, ABS.Keygen, ABS.Sign, ABS.Verify) consists of five probabilistic polynomial-time (PPT) algorithms:

- $\mathsf{TSetup}(\lambda, \mathcal{P}, n)$: is the randomized *trusted setup* algorithm taking as input a security parameter λ, an attribute universe $\mathcal{P}$ and an integer $n \in \mathsf{poly}(\lambda)$ which is an upper bound on the size of threshold policies. It outputs a set of public parameters pms (which contains λ, $\mathcal{P}$ and n). An execution of this algorithm is denoted as $\mathsf{pms} \leftarrow \mathsf{ABS.TSetup}(1^\lambda, \mathcal{P}, n)$.
- $\mathsf{MSetup}(\mathsf{pms})$: is the randomized *master setup* algorithm, that takes as input pms and outputs a master secret key msk and the corresponding master public key mpk.We write $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.MSetup}(\mathsf{pms})$ to denote an execution of this algorithm.
- $\mathsf{Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$: is a *key extraction* algorithm that takes in public parameters pms, the master keys mpk and msk, and an attribute set $\Omega \subset \mathcal{P}$. The output is a private key SK_Ω. To denote an execution of this algorithm, we write $SK_\Omega \leftarrow \mathsf{ABS.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$.
- $\mathsf{Sign}(\mathsf{pms}, \mathsf{mpk}, SK_\Omega, \mathsf{Msg}, \Gamma)$: is a randomized *signing* algorithm which takes as input the public parameters pms, the master public key mpk, a secret key SK_Ω, a message Msg and a threshold signing policy $\Gamma = (t, S)$ where $S \subset \mathcal{P}$ and $1 \leq t \leq |S| \leq n$. It outputs a signature σ. We denote the action taken by the signing algorithm as $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{pms}, \mathsf{mpk}, SK_\Omega, \mathsf{Msg}, \Gamma)$.
- $\mathsf{Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$: is a deterministic *verification* algorithm taking as input the public parameters pms, a master public key mpk, a message Msg, a signature σ and a threshold predicate $\Gamma = (t, S)$. It outputs 1 if the signature is deemed valid and 0 otherwise. To refer to an execution of the verification protocol we write $b \leftarrow \mathsf{ABS.Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$.

For correctness, for any $\lambda \in \mathbb{N}$, any integer $n \in \mathsf{poly}(\lambda)$, any universe $\mathcal{P}$, any set of public parameters $\mathsf{pms} \leftarrow \mathsf{ABS.TSetup}(1^\lambda, \mathcal{P}, n)$, any master key pair $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.MSetup}(\mathsf{pms})$, any subset $\Omega \subset \mathcal{P}$ and any threshold policy $\Gamma = (t, S)$ where $1 \leq t \leq |S|$, it is required that

$$\mathsf{ABS.Verify}\big(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \mathsf{ABS.Sign}(\mathsf{pms}, \mathsf{mpk}, SK_\Omega, \mathsf{Msg}, \Gamma), \Gamma\big) \;=\; 1$$

whenever $SK_\Omega \leftarrow \mathsf{ABS.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$ and $|\Omega \cap S| \geq t$.

2.4 Security of Threshold Attribute-Based Signatures

Unforgeability and privacy are the typical requirements for attribute-based signature schemes.

Unforgeability. An ABS scheme must satisfy the usual property of unforgeability, even against a group of colluding users that pool their secret keys. We consider a relaxed notion where the attacker *selects* the signing policy $\Gamma^\star = (t^\star, S^\star)$ that he wants to attack at the beginning of the game. However, the message $\mathsf{Msg}^\star$ whose signature is eventually forged is not selected in advance. The attacker can ask for valid signatures for messages and signing policies of his adaptive choice. The resulting property of *selective-predicate and adaptive-message unforgeability under chosen message attacks* ($\mathsf{sP\text{-}UF\text{-}CMA}$, for short) is defined by considering the following game.

Definition 4. *Let λ be an integer. Consider the following game between a probabilistic polynomial time (PPT) adversary $\mathcal{F}$ and its challenger.*

Initialization. *The challenger begins by specifying a universe of attributes $\mathcal{P}$ as well as an integer $n \in \mathsf{poly}(\lambda)$, which are sent to $\mathcal{F}$. Then, $\mathcal{F}$ selects a subset $S^\star \subset \mathcal{P}$ of attributes such that $|S^\star| \leq n$ and a threshold $t^\star \in \{1, \ldots, |S^\star|\}$. These define a threshold predicate $\Gamma^\star = (t^\star, S^\star)$.*

Setup. *The challenger runs $\mathsf{pms} \leftarrow \mathit{ABS.TSetup}(1^\lambda, \mathcal{P}, n)$ and $(\mathsf{mpk}, \mathit{msk}) \leftarrow \mathit{ABS.MSetup}(\mathsf{pms})$, and sends $\mathsf{pms}, \mathsf{mpk}$ to the forger $\mathcal{F}$.*

Queries. *$\mathcal{F}$ can interleave private key and signature queries.*

- ***Private key queries.*** *$\mathcal{F}$ adaptively chooses a subset of attributes $\Omega \subset \mathcal{P}$ under the restriction that $|\Omega \cap S^\star| < t^\star$ and must receive $SK_\Omega \leftarrow \mathit{ABS.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$ as the answer.*
- ***Signature queries.*** *$\mathcal{F}$ adaptively chooses a pair (Msg, Γ) consisting of a message Msg and a threshold predicate $\Gamma = (t, S)$ such that $1 \leq t \leq |S| \leq n$. The challenger chooses an arbitrary attribute set $\Omega \subset \mathcal{P}$ such that $|\Omega \cap S| \geq t$, runs $SK_\Omega \leftarrow \mathit{ABS.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$ and computes[2] a signature $\sigma \leftarrow \mathit{ABS.Sign}(\mathsf{pms}, \mathsf{mpk}, SK_\Omega, \mathsf{Msg}, \Gamma)$ which is returned to $\mathcal{F}$.*

Forgery. *At the end of the game, $\mathcal{F}$ outputs a pair $(\mathsf{Msg}^\star, \sigma^\star)$. We say that $\mathcal{F}$ is successful if:*

- *$\mathit{ABS.Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}^\star, \sigma^\star, \Gamma^\star) = 1$, and*
- *$\mathcal{F}$ has not made any signature query for the pair $(\mathsf{Msg}^\star, \Gamma^\star)$.*

The forger's advantage in breaking the sP-UF-CMA security of the scheme is defined as $\mathsf{Succ}^{\text{sP-UF-CMA}}_{\mathcal{F},\mathit{ABS}}(\lambda) = \Pr[\mathcal{F}\ \mathit{wins}]$. A threshold attribute-based signature scheme ABS is selective-predicate adaptive-message unforgeable (or sP-UF-CMA unforgeable) *if, for any PPT adversary $\mathcal{F}$, $\mathsf{Succ}^{\text{sP-UF-CMA}}_{\mathcal{F},\mathit{ABS}}(\lambda)$ is a negligible function of λ.*

Privacy (of Involved Attributes). This property ensures that a signature leaks nothing about the attributes that have been used to produce it beyond the fact that they satisfy the signing predicate. Privacy must hold even against attackers that control the master entity and is defined *via* a game between an adversary $\mathcal{D}$ and its challenger. Depending on the resources allowed to $\mathcal{D}$ and on its success probability, we can define computational privacy and perfect (unconditional) privacy.

Definition 5. *Let $\lambda \in \mathbb{N}$ and consider this game between a distinguisher $\mathcal{D}$ and its challenger.*

Setup. *The adversary $\mathcal{D}$ specifies a universe of attributes $\mathcal{P}$ and an integer $n \in \mathsf{poly}(\lambda)$, that are sent to the challenger. The challenger runs $\mathsf{pms} \leftarrow$*

[2] Since a given attribute set Ω may have many valid private keys SK_Ω, a generalization of the definition could allow $\mathcal{F}$ to obtain many signatures from the same private key SK_Ω. However, due to the signer privacy requirement, which is formalized hereafter, this does not matter.

$\mathit{ABS.TSetup}(1^\lambda, \mathcal{P}, n)$ *and sends* pms *to* $\mathcal{D}$. *The adversary* $\mathcal{D}$ *runs* $(\mathsf{mpk}, \mathit{msk}) \leftarrow \mathit{ABS.MSetup}(\mathsf{pms})$ *and sends* $(\mathsf{mpk}, \mathit{msk})$ *to the challenger (who must verify consistency of this master key pair).*

Challenge. $\mathcal{D}$ *outputs a tuple* $(\Gamma, \Omega_0, \Omega_1, \mathsf{Msg})$, *where* $\Gamma = (t, S)$ *is a threshold predicate such that* $1 \leq t \leq |S| \leq n$ *and* Ω_0, Ω_1 *are attribute sets satisfying* $|\Omega_b \cap S| \geq t$ *for each* $b \in \{0,1\}$. *The challenger picks a random bit* $\beta \stackrel{R}{\leftarrow} \{0,1\}$, *runs* $SK_{\Omega_\beta} \leftarrow \mathit{ABS.Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega_\beta)$ *and computes* $\sigma^\star \leftarrow \mathit{ABS.Sign}(\mathsf{pms}, \mathsf{mpk}, SK_{\Omega_\beta}, \mathsf{Msg}, \Gamma)$, *which is sent as a challenge to* $\mathcal{A}$.

Guess. $\mathcal{D}$ *outputs a bit* $\beta' \in \{0,1\}$ *and wins if* $\beta' = \beta$.

The advantage of $\mathcal{D}$ *is measured in the usual way, as the distance* $\mathsf{Adv}^{\mathit{Priv}}_{\mathcal{D},\mathit{ABS}}(\lambda) := |\Pr[\beta' = \beta] - \frac{1}{2}|$.

A threshold attribute-based signature scheme ABS *is said* computationally private *if* $\mathsf{Adv}^{\mathit{Priv}}_{\mathcal{D},\mathit{ABS}}(\lambda)$ *is a negligible function of* λ *for any PPT distinguisher* $\mathcal{D}$ *and it is said* perfectly/unconditionally private *if* $\mathsf{Adv}^{\mathit{Priv}}_{\mathcal{D},\mathit{ABS}}(\lambda) = 0$ *for any (possibly computationally unbounded) distinguisher* $\mathcal{D}$.

3 A First Short Attribute-Based Signature Scheme for Threshold Predicates

We present here our first scheme to produce attribute-based signatures with constant size, for threshold predicates. The secret key sk_Ω for a user holding a set of attributes Ω contains $|\Omega|+n$ elements, where n is the maximum size of the attribute set for any signing policy. This construction is for "small" universes of attributes $\mathcal{P} = \{\mathsf{at}_1, \ldots, \mathsf{at}_\eta\}$, for some integer $\eta \in \mathbb{N}$, as public parameters have linear size in η; therefore, η must be polynomial in the security parameter of the scheme. Attributes $\{\mathsf{at}_i\}_{i=1}^{\eta}$ are arbitrary strings which some encoding function ς maps to $\mathbb{Z}_p^*$. Since the scheme is a small universe construction, we may set $n = \eta$ in the description hereafter.

The construction builds on the ABE scheme of Herranz *et al.* [17]. The intuition is to have the signer implicitly prove his ability to decrypt a ciphertext corresponding to that ABE scheme. This non-interactive proof is generated using the Groth-Sahai proof systems [16], by binding the signed message (and the corresponding predicate) to the non-interactive proof using a technique suggested by Malkin *et al.* [23]. In some sense, this technique can be seen as realizing signatures of knowledge in the standard model: it consists in embedding the message to be signed in the Groth-Sahai CRS by calculating part of the latter as a "hash value" of the message. As noted in [23], Waters' hash function [32] is well-suited to this purpose since, in the security proof, it makes it possible to answer signing queries using simulated NIZK proofs. At the same time, with non-negligible probability, adversarially-generated signatures are produced using a perfectly sound Groth-Sahai CRS and they thus constitute real proofs, from which witnesses can be extracted.

In [23], the above technique was applied to an instantiation of Groth-Sahai proofs based on the Symmetric eXternal Diffie-Hellman assumption (and thus

asymmetric pairings). In this section, we adapt this technique so as to get it to work with symmetric pairings and the linear assumption.

In the notations of the verification algorithm, when $\boldsymbol{C} = (C_1, C_2, C_3)^\top \in \mathbb{G}^3$ is a vector of group elements and if $g \in \mathbb{G}$, we denote by $E(g, \boldsymbol{C})$ the vector of pairing values $\big(e(g, C_1), e(g, C_2), e(g, C_3)\big)^\top$.

▶ $\mathsf{TSetup}(\lambda, \mathcal{P}, n)$: the trusted setup algorithm conducts the following steps.

1. Choose groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Select generators $g, h \xleftarrow{R} \mathbb{G}$ and also choose a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^k$, for some $k \in \mathsf{poly}(\lambda)$.
2. Define a suitable injective encoding ς sending each one of the n attributes $\mathsf{at} \in \mathcal{P}$ onto an element $\varsigma(\mathsf{at}) = x \in \mathbb{Z}_p^\star$. Choose a set $\mathcal{D} = \{d_1, \ldots, d_{n-1}\}$ consisting of $n-1$ pairwise different elements of $\mathbb{Z}_p^*$, which must also be different from the encoding of any attribute in $\mathcal{P}$. For any integer i lower or equal to $n-1$, we denote as $\mathcal{D}_i$ the set $\{d_1, \ldots, d_i\}$.
3. Generate Groth-Sahai reference strings by choosing random generators $g_1, g_2 \xleftarrow{R} \mathbb{G}$ and defining vectors $\boldsymbol{g_1} = (g_1, 1, g)^\top \in \mathbb{G}^3$ and $\boldsymbol{g_2} = (1, g_2, g)^\top \in \mathbb{G}^3$. Then, for each $i \in \{0, \ldots, k\}$, pick $\xi_{i,1}, \xi_{i,2} \xleftarrow{R} \mathbb{Z}_p$ at random and define a vector $\boldsymbol{g_{3,i}} = \boldsymbol{g_1}^{\xi_{i,1}} \cdot \boldsymbol{g_2}^{\xi_{i,2}} = \big(g_1^{\xi_{i,1}}, g_2^{\xi_{i,2}}, g^{\xi_{i,1}+\xi_{i,2}}\big)^\top$. Exponents $\{(\xi_{i,1}, \xi_{i,2})\}_{i=0}^k$ can then be discarded as they are no longer needed.

The resulting public parameters are

$$\mathsf{pms} = \Big(\mathcal{P},\ n,\ \lambda,\ \mathbb{G},\ \mathbb{G}_T,\ g,\ h,\ \boldsymbol{g_1},\ \boldsymbol{g_2},\ \{\boldsymbol{g_{3,i}}\}_{i=0}^k,\ H,\ \varsigma,\ \mathcal{D}\Big).$$

▶ $\mathsf{MSetup}(\mathsf{pms})$: picks at random $\alpha, \gamma \xleftarrow{R} \mathbb{Z}_p^*$ and sets $u = g^{\alpha\gamma}$ and $v = e(g^\alpha, h)$. The master secret key is $\mathsf{msk} = (\alpha, \gamma)$ and the master public key consists of

$$\mathsf{mpk} = \Big(\ u,\ v,\ g^\alpha,\ \Big\{h^{\alpha\gamma^i}\Big\}_{i=0,\ldots,2n-1}\Big).$$

▶ $\mathsf{Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$: given an attribute set Ω and $\mathsf{msk} = (\alpha, \gamma)$, pick $r \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$SK_\Omega = \Big(\Big\{g^{\frac{r}{\gamma+\varsigma(\mathsf{at})}}\Big\}_{\mathsf{at}\in\Omega}, \Big\{h^{r\gamma^i}\Big\}_{i=0,\ldots,n-2}, h^{\frac{r-1}{\gamma}}\Big). \tag{2}$$

▶ $\mathsf{Sign}(\mathsf{pms}, \mathsf{mpk}, SK_\Omega, \mathsf{Msg}, \Gamma)$: to sign $\mathsf{Msg} \in \{0,1\}^*$ w.r.t. the policy $\Gamma = (t, S)$, where $S \subset \mathcal{P}$ is an attribute set of size $s = |S| \leq n$ and $1 \leq t \leq s \leq n$, the algorithm returns $\perp$ if $|\Omega \cap S| < t$. Otherwise, it first parses SK_Ω as in (2) and conducts the following steps.

1. Let Ω_S be any subset of $\Omega \cap S$ with $|\Omega_S| = t$. From all $\mathsf{at} \in \Omega_S$, using the algorithm $\mathsf{Aggregate}$ of [12], compute the value

$$A_1 = \mathsf{Aggregate}(\{g^{\frac{r}{\gamma+\varsigma(\mathsf{at})}}, \varsigma(\mathsf{at})\}_{\mathsf{at}\in\Omega_S}) = g^{\frac{r}{\prod_{\mathsf{at}\in\Omega_S}(\gamma+\varsigma(\mathsf{at}))}}.$$

From A_1, compute $T_1 = A_1^{\frac{1}{\prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})\setminus\Omega_S}\varsigma(\mathsf{at})}}$.

2. Define the value $P_{(\Omega_S,S)}(\gamma)$ as

$$P_{(\Omega_S,S)}(\gamma) = \frac{1}{\gamma}\left(\prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})\setminus\Omega_S}(\gamma+\varsigma(\mathsf{at})) - \prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})\setminus\Omega_S}\varsigma(\mathsf{at})\right).$$

Since $|\Omega_S| = t$, the degree of $P_{(\Omega_S,S)}(X)$ is $n-2$. Therefore, from the private key SK_Ω, one can compute $h^{r\cdot P_{(\Omega_S,S)}(\gamma)/(\prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})\setminus\Omega_S}\varsigma(\mathsf{at}))}$ and multiply it with the last element $h^{\frac{r-1}{\gamma}}$ of SK_Ω to obtain

$$T_2 = h^{\frac{r-1}{\gamma}}\cdot h^{r\frac{P_{(\Omega_S,S)}(\gamma)}{\prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})\setminus\Omega_S}\varsigma(\mathsf{at})}}.$$

Note that the obtained values $T_1, T_2 \in \mathbb{G}$ satisfy the equality

$$e(T_2,u^{-1})\cdot e\Big(T_1, h^{\alpha\cdot\prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})}(\gamma+\varsigma(\mathsf{at}))}\Big) = e(g^\alpha,h) \qquad (3)$$

and that, in the terms in the left-hand-side of equality (3), the second argument of each pairing is publicly computable using pms and mpk.

3. Compute $M = m_1\ldots m_k = H(\mathsf{Msg},\Gamma) \in \{0,1\}^k$ and use M to form a message-specific Groth-Sahai CRS $\mathbf{g}_M = (\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g}_{3,M})$. Namely, for $i = 0$ to k, parse $\boldsymbol{g_{3,i}}$ as $(g_{X,i}, g_{Y,i}, g_{Z,i})^\top \in \mathbb{G}^3$. Then, define the vector $\boldsymbol{g}_{3,M} = \big(g_{X,0}\cdot\prod_{i=1}^k g_{X,i}^{m_i},\ g_{Y,0}\cdot\prod_{i=1}^k g_{Y,i}^{m_i},\ g_{Z,0}\cdot\prod_{i=1}^k g_{Z,i}^{m_i}\big)^\top$.
4. Using the newly defined $\mathbf{g}_M = (\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g}_{3,M})$, generate Groth-Sahai commitments to T_1 and T_2. Namely, pick $r_1, s_1, t_1, r_2, s_2, t_2 \xleftarrow{R} \mathbb{Z}_p$ and compute $\boldsymbol{C}_{T_j} = (1,1,T_j)^\top \cdot \boldsymbol{g_1}^{r_j}\cdot \boldsymbol{g_2}^{s_j}\cdot \boldsymbol{g}_{3,M}^{t_j}$ for $j \in \{1,2\}$. Then, generate a NIZK proof that committed variables (T_1, T_2) satisfy the pairing-product equation (3). To this end, we introduce an auxiliary variable $\Theta \in \mathbb{G}$ (with its own commitment $\boldsymbol{C}_\Theta = (1,1,\Theta)^\top \cdot \boldsymbol{g_1}^{r_\theta}\cdot \boldsymbol{g_2}^{s_\theta}\cdot \boldsymbol{g}_{3,M}^{t_\theta}$, for $r_\theta, s_\theta, t_\theta \xleftarrow{R} \mathbb{Z}_p$), which takes on the value $\Theta = h$, and actually prove that

$$e(T_1,H_S) = e(g^\alpha,\Theta)\cdot e(T_2,u) \qquad (4)$$
$$e(g,\Theta) = e(g,h), \qquad (5)$$

where $H_S = h^{\alpha\cdot\prod_{\mathsf{at}\in(S\cup\mathcal{D}_{n+t-1-s})}(\gamma+\varsigma(\mathsf{at}))}$. The proofs for relations (4) and (5) are called $\boldsymbol{\pi_1}$ and $\boldsymbol{\pi_2}$, respectively, and they are given by

$$\boldsymbol{\pi}_1 = \big(H_S^{r_1}\cdot(g^\alpha)^{-r_\theta}\cdot u^{-r_2},\ H_S^{s_1}\cdot(g^\alpha)^{-s_\theta}\cdot u^{-s_2},\ H_S^{t_1}\cdot(g^\alpha)^{-t_\theta}\cdot u^{-t_2}\big)^\top$$
$$\boldsymbol{\pi}_2 = \big(g^{r_\theta},\ g^{s_\theta},\ g^{t_\theta}\big)^\top.$$

Finally, output the signature $\sigma = \big(\boldsymbol{C}_{T_1}, \boldsymbol{C}_{T_2}, \boldsymbol{C}_\theta, \boldsymbol{\pi_1}, \boldsymbol{\pi_2}\big) \in \mathbb{G}^{15}$.

▶ $\mathsf{Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$: it first parses Γ as a pair (t, S) and σ as $\big(\boldsymbol{C}_{T_1}, \boldsymbol{C}_{T_2}, \boldsymbol{C}_\theta, \boldsymbol{\pi_1}, \boldsymbol{\pi_2}\big)$. It computes $M = m_1 \dots m_k = H(\mathsf{Msg}, \Gamma) \in \{0,1\}^k$ and forms the corresponding vector

$$\boldsymbol{g}_{3,M} = \Big(g_{X,0} \cdot \prod_{i=1}^{k} g_{X,i}^{m_i},\ g_{Y,0} \cdot \prod_{i=1}^{k} g_{Y,i}^{m_i},\ g_{Z,0} \cdot \prod_{i=1}^{k} g_{Z,i}^{m_i}\Big)^\top \in \mathbb{G}^3.$$

Then, parse the proofs $\boldsymbol{\pi_1}$ and $\boldsymbol{\pi_2}$ as vectors $(\pi_{1,1}, \pi_{1,2}, \pi_{1,3})^\top$ and $(\pi_{2,1}, \pi_{2,2}, \pi_{2,3})^\top$, respectively. Define $H_S = h^{\alpha \cdot \prod_{\mathsf{at} \in (S \cup \mathcal{D}_{n+t-1-s})} (\gamma + \varsigma(\mathsf{at}))}$ and return 1 if and only if these relations are both satisfied:

$$E(H_S, \boldsymbol{C_{T_1}}) = E(g^\alpha, \boldsymbol{C_\theta}) \cdot E(u, \boldsymbol{C_{T_2}}) \cdot E(\pi_{1,1}, \boldsymbol{g_1}) \cdot E(\pi_{1,2}, \boldsymbol{g_2}) \cdot E(\pi_{1,3}, \boldsymbol{g}_{3,M}) \quad (6)$$

$$E(g, \boldsymbol{C_\theta}) = E\big(g, (1,1,h)\big) \cdot E(\pi_{2,1}, \boldsymbol{g_1}) \cdot E(\pi_{2,2}, \boldsymbol{g_2}) \cdot E(\pi_{2,3}, \boldsymbol{g}_{3,M}). \quad (7)$$

Correctness. The correctness follows from that of Groth-Sahai proofs.

Security Analysis. The scheme is selective-predicate and adaptive-message unforgeable assuming the hardness of both the DLIN problem and the $(\tilde{\ell}, \tilde{m}, \tilde{t})$-aMSE-CDH problem. Computational privacy can be proven based on the hardness of the DLIN problem.

Theorem 1. *The scheme is selective-predicate and adaptive-message unforgeable under chosen-message attacks assuming that (1) H is a collision-resistant hash function; (2) the DLIN assumption holds in $\mathbb{G}$; (3) the $(\tilde{\ell}, \tilde{m}, \tilde{t})$-aMSE-CDH assumption holds in $(\mathbb{G}, \mathbb{G}_T)$.* (The proof can be found in [18]).

Theorem 2. *This scheme has computational privacy, assuming that DLIN holds in $\mathbb{G}$.*

Proof. (Sketch.) The proof consists in considering two games: Game_0 and Game_1. The first game, Game_0, is the real privacy game as described in Definition 5. In particular, when executing the trusted setup algorithm ABS.TSetup, the challenger chooses the vectors $(\boldsymbol{g_1}, \boldsymbol{g_2}, \{\boldsymbol{g_{3,i}}\}_{i=0}^k)$ such that $\boldsymbol{g_{3,i}}$ is linearly dependent with $(\boldsymbol{g_1}, \boldsymbol{g_2})$, for all $i = 0, \dots, k$. The only difference between Game_1 and Game_0 is that, in Game_1, the vector $\boldsymbol{g_{3,i}}$ is chosen at random so that it is linearly independent with $(\boldsymbol{g_1}, \boldsymbol{g_2})$, for all $i = 0, \dots, k$. Groth-Sahai [16] proved that this change is indistinguishable, under the DLIN assumption. Finally, in Game_1, the only values that could leak any information about the subset of attributes held by the signer are $\boldsymbol{C}_{T_1}, \boldsymbol{C}_{T_2}, \boldsymbol{\pi}_1$. But in the setting of Game_1, these commitments and proofs are perfectly hiding: they do not reveal any information about the committed values T_1, T_2. Therefore, privacy of the attributes holds unconditionally in Game_1. □

4 A Second Short Attribute-Based Signature Scheme for Threshold Predicates

The main advantage of our second ABS scheme over the previous one is that signatures are much shorter, since they have only three group elements. This

comes at the cost of longer secret keys sk_Ω, containing $(2n+2) \times (|\Omega| + n)$ group elements. Another advantage is that the size of the considered universe of attributes may be much larger, even exponential in the security parameter λ; we only need that all attributes in the universe $\mathcal{P}$ are encoded as different elements of $\mathbb{Z}_p^*$.

▶ $\mathsf{TSetup}(\lambda, \mathcal{P}, n)$: chooses a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^k$, for some integer $k \in \mathsf{poly}(\lambda)$, as well as bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with $g \xleftarrow{R} \mathbb{G}$. It also picks $u_0, u_1, \ldots, u_k \xleftarrow{R} \mathbb{G}$ and sets $\boldsymbol{U} = (u_0, u_1, \ldots, u_k)^\top$. It finally chooses a set $\mathcal{D} = \{d_1, \ldots, d_n\}$ of n distinct elements of $\mathbb{Z}_p$ that will serve as dummy attributes.
The resulting public parameters are $\mathsf{pms} = \Big(\mathcal{P},\ n,\ \lambda,\ \mathbb{G},\ \mathbb{G}_T, g,\ \boldsymbol{U},\ \mathcal{D},\ H\Big)$.

▶ $\mathsf{MSetup}(\mathsf{pms})$: randomly chooses $\alpha, \alpha_0 \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^\top \xleftarrow{R} \mathbb{Z}_p^N$, where $N = 2n+1$. It then computes $e(g,g)^\alpha$, $h_0 = g^{\alpha_0}$, $\boldsymbol{H} = (h_1, \ldots, h_N)^\top = g^{\boldsymbol{\alpha}}$. The master secret key is defined to be $\mathsf{msk} = g^\alpha$ and the master public key is $\mathsf{mpk} = \Big(e(g,g)^\alpha,\ h_0,\ \boldsymbol{H}\Big)$.

▶ $\mathsf{Keygen}(\mathsf{pms}, \mathsf{mpk}, \mathsf{msk}, \Omega)$: to generate a key for the attribute set Ω, the algorithm picks a polynomial $Q_\Omega[X] = \alpha + \beta_1 X + \cdots + \beta_{n-1} X^{n-1}$ where $\beta_1, \ldots, \beta_{n-1} \xleftarrow{R} \mathbb{Z}_p$. Then, it proceeds as follows.

1. For each attribute $\omega \in \Omega$, choose a random exponent $r_\omega \xleftarrow{R} \mathbb{Z}_p$ and generate a key component $\mathsf{SK}_\omega = (D_{\omega,1}, D_{\omega,2}, K_{\omega,1}, \ldots, K_{\omega,N-1})$ where

$$D_{\omega,1} = g^{Q_\Omega(\omega)} \cdot h_0^{r_\omega}, \quad D_{\omega,2} = g^{r_\omega}, \quad \Big\{K_{\omega,i} = \big(h_1^{-\omega^i} \cdot h_{i+1}\big)^{r_\omega}\Big\}_{i=1,\ldots,N-1}. \quad (8)$$

2. For each $d \in \mathcal{D}$, choose a fresh random value $r_d \in \mathbb{Z}_p$ and generate a private key component $\mathsf{SK}_d = (D_{d,1}, D_{d,2}, K_{d,1}, \ldots, K_{d,N-1})$ as in (8):

$$D_{d,1} = g^{Q_\Omega(d)} \cdot h_0^{r_d}, \quad D_{d,2} = g^{r_d}, \quad \Big\{K_{d,i} = \big(h_1^{-w^i} \cdot h_{i+1}\big)^{r_d}\Big\}_{i=1,\ldots,N-1}. \quad (9)$$

The private key finally consists of $SK_\Omega = \big(\{\mathsf{SK}_\omega\}_{\omega\in\Omega}, \{\mathsf{SK}_d\}_{d\in\mathcal{D}}\big)$.

▶ $\mathsf{Sign}(\mathsf{pms}, \mathsf{mpk}, SK_\Omega, \mathsf{Msg}, \Gamma)$: to sign $\mathsf{Msg} \in \{0,1\}^*$ w.r.t. the policy $\Gamma = (t, S)$, where S is an attribute set of size $s = |S| \le n$ and $t \in \{1, \ldots, s\}$, the algorithm first computes $M = H(\mathsf{Msg}, \Gamma) \in \{0,1\}^k$ and parses the private key SK_Ω as $\big(\{\mathsf{SK}_\omega\}_{\omega\in\Omega}, \{\mathsf{SK}_d\}_{d\in\mathcal{D}}\big)$.

1. It considers the subset $\mathcal{D}_{n-t} \subset \mathcal{D}$ containing the $n-t$ first attributes of $\mathcal{D}$ (chosen in some pre-specified lexicographical order). It also chooses an arbitrary subset $S_t \subset \Omega \cap S$ such that $|S_t| = t$ and defines $\boldsymbol{Y} = (y_1, \ldots, y_N)^\top$ as the vector containing the coefficients of the polynomial

$$P_S(Z) = \sum_{i=1}^{n-t+s+1} y_i Z^{i-1} = \prod_{\omega\in S}(Z-\omega) \cdot \prod_{d\in\mathcal{D}_{n-t}} (Z-d). \quad (10)$$

Since $n-t+s+1 \leq 2n+1 = N$, the coordinates $y_{n-t+s+2}, \ldots, y_N$ are all set to 0.

2. For each $\omega \in S_t$, use $\mathsf{SK}_\omega = (D_{\omega,1}, D_{\omega,2}, \{K_{\omega,i}\}_{i=1}^{N-1})$ to compute
$$D'_{\omega,1} = D_{\omega,1} \cdot \prod_{i=1}^{N-1} K_{\omega,i}^{y_{i+1}} = g^{Q_\Omega(\omega)} \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^{r_\omega}. \tag{11}$$
The last equality comes from the fact that $P_S(\omega) = 0$ for all $\omega \in S$.
3. Likewise, for each dummy attribute $d \in \mathcal{D}_{n-t}$, use $\mathsf{SK}_d = (D_{d,1}, D_{d,2}, \{K_{d,i}\}_{i=1}^{N-1})$ to compute
$$D'_{d,1} = D_{d,1} \cdot \prod_{i=1}^{N-1} K_{d,i}^{y_{i+1}} = g^{Q_\Omega(d)} \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^{r_d}. \tag{12}$$
4. Using $\{D'_{\omega,1}\}_{\omega \in S_t}$ and $\{D'_{d,1}\}_{d \in \mathcal{D}_{n-t}}$ and the corresponding $D_{\omega,2} = g^{r_\omega}$, $D_{d,2} = g^{r_d}$, compute
$$D_1 = \prod_{\omega \in S_t} D'_{\omega,1}{}^{\Delta_\omega^{S_t \cup \mathcal{D}_{n-t}}(0)} \cdot \prod_{d \in \mathcal{D}_{n-t}} D'_{d,1}{}^{\Delta_d^{S_t \cup \mathcal{D}_{n-t}}(0)} = g^\alpha \cdot (h_0 \cdot \prod_{i=1}^{N} h_i^{y_i})^r \tag{13}$$
$$D_2 = \prod_{\omega \in S_t} D_{\omega,2}{}^{\Delta_\omega^{S_t \cup \mathcal{D}_{n-t}}(0)} \cdot \prod_{d \in \mathcal{D}_{n-t}} D_{d,2}{}^{\Delta_d^{S_t \cup \mathcal{D}_{n-t}}(0)} = g^r, \tag{14}$$
where $r = \sum_{\omega \in S_t} \Delta_\omega^{S_t \cup \mathcal{D}_{n-t}}(0) \cdot r_\omega + \sum_{d \in \mathcal{D}_{n-t}} \Delta_d^{S_t \cup \mathcal{D}_{n-t}}(0) \cdot r_d$.
5. Parse $M \in \{0,1\}^k$ as a string $m_1 \ldots m_k$ where $m_j \in \{0,1\}$ for $j = 1, \ldots, k$. Then, choose $z, w \xleftarrow{R} \mathbb{Z}_p$ and compute
$$\sigma_1 = D_1 \cdot \big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^w \cdot \big(u_0 \cdot \prod_{j=1}^{k} u_j^{m_j}\big)^z, \qquad \sigma_2 = D_2 \cdot g^w, \qquad \sigma_3 = g^z.$$

Return the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \mathbb{G}^3$.

▶ $\mathsf{Verify}(\mathsf{pms}, \mathsf{mpk}, \mathsf{Msg}, \sigma, \Gamma)$: it parses Γ as a pair (t, S). It computes $M = H(\mathsf{Msg}, \Gamma) \in \{0,1\}^k$ and considers the subset $\mathcal{D}_{n-t} \subset \mathcal{D}$ containing the $n-t$ first dummy attributes of $\mathcal{D}$. Then, it defines the vector $\boldsymbol{Y} = (y_1, \ldots, y_N)^\top$ from the polynomial $P_S(Z)$ as per (10). The algorithm accepts the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ as valid and thus outputs 1 if and only if
$$e(g,g)^\alpha = e(\sigma_1, g) \cdot e\big(\sigma_2, h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\big)^{-1} \cdot e\big(\sigma_3, u_0 \cdot \prod_{j=1}^{k} u_j^{m_j}\big)^{-1}. \tag{15}$$

Correctness. The correctness of the scheme follows from the property that for each attribute $\omega \in S_t \subset S \cap \Omega$, the vector $\boldsymbol{X}_\omega^N = (1, \omega, \omega^2, \ldots, \omega^{N-1})$ is orthogonal to $\boldsymbol{Y}$, so that we have
$$D'_{\omega,1} = g^{Q_\Omega(\omega)} \cdot \Big(h_0 \cdot h_1^{-(\langle \boldsymbol{X}_\omega^N, \boldsymbol{Y} \rangle - y_1)} \prod_{i=2}^{N} h_i^{y_i}\Big)^{r_\omega} = g^{Q_\Omega(\omega)} \cdot \Big(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}\Big)^{r_\omega},$$

which explains the second equality of (11) and the same holds for (12). In addition, the values (D_1, D_2) obtained as per (13)-(14) satisfy $e(D_1, g) = e(g, g)^{\alpha} \cdot e(h_0 \cdot \prod_{i=1}^{N} h_i^{y_i}, D_2)$, which easily leads to the verification equation (15).

Security Analysis. This second scheme is selective-predicate and adaptive-message unforgeable by reduction to the hardness of the n-Diffie-Hellman Exponent (n-DHE) problem ([6]). This scheme also enjoys unconditional privacy, which is another advantage over our first scheme.

Theorem 3. *The scheme is selective-predicate and adaptive-message unforgeable under chosen-message attacks if H is collision-resistant and if the $(2n+1)$-DHE assumption holds in $\mathbb{G}$, where n is the maximal number of attributes in the set S.* (The proof can be found in [18].)

Theorem 4. *This second ABS scheme enjoys perfect privacy.*

Proof. A valid signature for the threshold policy (t, S) which was produced using the subset of attributes $S_t \subset S$, $|S_t| = t$ and with randomness w can also be produced for any other set $S'_t \subset S$, $|S'_t| = t$ with randomness w'. More specifically, if $r = \sum_{\omega \in S_t} \Delta_{\omega}^{S_t \cup \mathcal{D}_{n-t}}(0) \cdot r_{\omega} + \sum_{d \in \mathcal{D}_{n-t}} \Delta_{d}^{S_t \cup \mathcal{D}_{n-t}}(0) \cdot r_d$ and $r' = \sum_{\omega \in S'_t} \Delta_{\omega}^{S_t \cup \mathcal{D}_{n-t}}(0) \cdot r_{\omega} + \sum_{d \in \mathcal{D}_{n-t}} \Delta_{d}^{S_t \cup \mathcal{D}_{n-t}}(0) \cdot r_d$, any pair (w, w') satisfying $r + w = r' + w'$ will result in the same signature for S_t and S'_t. □

5 More General Signing Predicates

Our schemes admit some extensions to deal with more general monotone predicates. In general, a predicate is a pair (S, Γ), where $S = \{\mathsf{at}_1, \ldots, \mathsf{at}_s\}$ is a set of attributes and $\Gamma \subset 2^S$ is a monotone increasing family of subsets of S. An attribute-based signature for a pair (S, Γ) convinces the verifier that the signer holds some subset of attributes $A \in \Gamma$, without revealing any information on A.

5.1 Extensions for the First Scheme

Similarly to what is suggested in [12], our first signature scheme can be extended to admit weighted threshold predicates, that is, pairs (S, Γ) for which there exists a threshold t and an assignment of weights $\omega : S \to \mathbb{Z}^+$ such that $\Omega \in \Gamma \iff \sum_{\mathsf{at} \in \Omega} \omega(\mathsf{at}) \geq t$.

Furthermore, since the final form of the signatures in our first threshold scheme is that of a Groth-Sahai non-interactive proof, one could consider signing predicates which are described by a monotone formula (OR / AND gates) over threshold clauses. The Groth-Sahai proof would be then a proof of knowledge of some values that satisfy a monotone formula of equations. The size of such a proof (and therefore, the size of the resulting attribute-based signatures) would be linear in the number of threshold clauses in the formula. We stress that this is still better than having size linear in the number of involved attributes, as in all previous constructions.

5.2 Extensions for the Second Scheme

The idea of our second scheme is that a (threshold) attribute-based signature can be computed only if the signer holds t attributes in S which, combined with $n - t$ dummy attributes, lead to n attributes at such that $P_S(\mathsf{at}) = 0$. This makes it possible to interpolate a polynomial $Q_\Omega(X)$ with degree $n - 1$, recover in some way the value g^α and produce a valid signature. To admit any possible value of the threshold t in $\{1, \ldots, n\}$, the number of dummy attributes must be n. We can use similar ideas for other families of predicates which are realized with a secret sharing scheme with properties which resemble those of Shamir's. The ideas underlying this extension are quite related to those in [11], where dummy attributes were used to design attribute-based encryption schemes for general decryption predicates. An illustrative example, considering hierarchical threshold predicates, is given in the full version of this paper [18].

Disclaimer and Acknowledgments. This work was started while the second and third authors visited Universitat Politècnica de Catalunya. J. Herranz is supported by a *Ramón y Cajal* grant, partially funded by the European Social Fund (ESF) of the Spanish MICINN Ministry, F. Laguillaumie by the French ANR-07-TCOM-013-04 PACE Project and B. Libert by the F.RS.-F.N.RS. through a "Chargé de recherches" fellowship and by the BCRYPT Interuniversity Attraction Pole. Both J. Herranz and C. Ràfols are partially supported by the Spanish MICINN Ministry under project MTM2009-07694 and ARES – CONSOLIDER INGENIO 2010 CSD2007-00004. C. Ràfols is with the UNESCO Chair in Data Privacy, but the views expressed in this paper are her own and do not commit UNESCO.

References

1. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive Key-Policy Attribute-Based Encryption with Constant-Size Ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE S&P 2007, pp. 321–334. IEEE Society Press (2007)
3. 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)
4. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity-Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
5. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
6. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)

7. Boneh, D., Hamburg, M.: Generalized Identity-Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008)
8. Boyen, X.: Mesh Signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
9. Bresson, E., Stern, J., Szydlo, M.: Threshold Ring Signatures and Applications to Ad-Hoc Groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002)
10. Chase, M., Lysyanskaya, A.: On Signatures of Knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)
11. Daza, V., Herranz, J., Morillo, P., Ràfols, C.: Extended access structures and their cryptographic applications. Applicable Algebra in Engineering, Communication and Computing 21(4), 257–284 (2010)
12. Delerablée, C., Pointcheval, D.: Dynamic Threshold Public-Key Encryption. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 317–334. Springer, Heidelberg (2008)
13. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009)
14. Escala, A., Herranz, J., Morillo, P.: Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 224–241. Springer, Heidelberg (2011)
15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98. ACM Press (2006)
16. Groth, J., Sahai, A.: Efficient Non-Interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
17. Herranz, J., Laguillaumie, F., Ràfols, C.: Constant-size Ciphertexts in Threshold Attribute-Based Encryption. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 19–34. Springer, Heidelberg (2010)
18. Herranz, J., Libert, B., Laguillaumie, F., Ràfols, C.: Short attribute-based signatures for threshold predicates (preprint) (2011), http://hal.archives-ouvertes.fr/hal-00611651/fr/
19. Hofheinz, D., Kiltz, E.: Programmable Hash Functions and their Applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
20. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
21. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: ASIACCS 2010, pp. 60–69. ACM Press (2010)
22. Li, J., Kim, K.: Hidden attribute-based signatures without anonymity revocation. Information Sciences 180(9), 1681–1689 (2010)
23. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures Resilient to Continual Leakage on Memory and Computation. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 89–106. Springer, Heidelberg (2011)

24. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-Based Signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011)
25. Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
26. Okamoto, T., Takashima, K.: Homomorphic Encryption and Signatures from Vector Decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)
27. Okamoto, T., Takashima, K.: Efficient Attribute-Based Signatures for Non-Monotone Predicates in the Standard Model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011)
28. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
29. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Shahandashti, S.F., Safavi-Naini, R.: Threshold Attribute-Based Signatures and their Application to Anonymous Credential Systems. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 198–216. Springer, Heidelberg (2009)
31. Tassa, T.: Hierarchical threshold secret sharing. Journal of Cryptology 20(2), 237–264 (2007)
32. Waters, B.: Efficient Identity-Based Encryption without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Reducing the Key Size of Rainbow Using Non-commutative Rings

Takanori Yasuda[1], Kouichi Sakurai[1,2], and Tsuyoshi Takagi[3]

[1] Institute of Systems, Information Technologies and Nanotechnologies
[2] Department of Informatics, Kyushu University
[3] Institute of Mathematics for Industry, Kyushu University

Abstract. Multivariate Public Key Cryptosystems (MPKC) are candidates for post-quantum cryptography. Rainbow is a digital signature scheme in MPKC, whose encryption and decryption are relatively efficient. However, the security of MPKC depends on the difficulty in solving a system of multivariate polynomials, and the key length of MPKC becomes substantially large compared with that of RSA cryptosystems for the same level of security. The size of the public key in MPKC has been reduced in previous research, but to the best of our knowledge, there are no algorithms to reduce the size of a private key . In this paper, we propose NC-Rainbow, a variation of Rainbow using non-commutative rings and we describe the ability of the proposed scheme to reduce the size of a private key in comparison with the ordinary Rainbow while maintaining the same level of security. In particular, using the proposed NC-Rainbow, the size of a private key is reduced by about 75% at the 80 bit security level. Moreover, the speed of signature generation is accelerated by about 34% at the 80 bit security level.

Keywords: Multivariate Public Key Cryptosystem, Digital signature, Rainbow, Non-commutative ring, Key size reduction, Post-quantum cryptography.

1 Introduction

Multivariate Public Key Cryptosystems (MPKC) [7] can be potentially applied to post-quantum cryptography. Rainbow is a digital signature scheme in MPKC that affords relatively efficient encryption and decryption [6]. However, the security of MPKC depends on the difficulty in solving a system of multivariate polynomials, and a substantial number of their coefficients is required to attain a reasonable level of security. Because the set of such coefficients is used for public and secret keys, the key size eventually increases. In fact, in the case of Rainbow$(GF(256); 17, 13, 13)$, which is as secure as a 1024 bit RSA signature scheme [19], the sizes of secret and public keys increase to about 150 and 200 times 1024 bits, respectively.

For a public key cryptosystem, it is important to study the reduction of key size. In the case of RSA with a small key size, the lattice attack works efficiently

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 68–83, 2012.

[26,5], whereas in the case of cryptosystems based on discrete logarithm, Pollard's λ-method [23,17] works efficiently [11]. Moreover, the key size of the McEliece cryptosystem, which is another candidate for post-quantum cryptography, has been reduced by Berger et al. [3]. In the case of Rainbow, it is known that CyclicRainbow ([18,20,21]) reduces the size of the public key while maintaining the security of the original Rainbow. However, to the best of our knowledge, research on reducing the secret key size of Rainbow has yet to be reported.

In this paper, we try to reduce the secret key size of Rainbow by using non-commutative rings. The proposed scheme is constructed by replacing a definition field with a non-commutative ring in the original Rainbow scheme. Non-commutative rings are a well-established topic in mathematics [10]; for examples, quaternion algebras and group rings have been studied in depth. A non-commutative ring can be embedded into an appropriate matrix ring, which yields a compact representation of elements. Therefore, if the matrices appearing in the description of the secret key of Rainbow are described through embedding, we are then able to obtain a compact representation of the secret key. Our proposed scheme can also be regarded as a new method for constructing Rainbow. In fact, for any construction of the proposed scheme, there is a corresponding Rainbow with an identical public key.

It is necessary to evaluate the security of the proposed scheme. There are six types of attacks that can occur against Rainbow [19], namely 1) direct, 2) UOV, 3) MinRank, 4) HighRank, 5) Rainbow-Band-Separation (RBS), and (6) UOV-Reconciliation (UOV-R) attacks. We estimate the security of the proposed scheme by applying these attacks. Our analysis determines that these attacks are unable to achieve a lower complexity and break the proposed scheme. Finally, combining this cryptanalysis with the secure parameter recommended [19], we present the parameters of the proposed non-commutative Rainbow at a security level of more than 80 bits. For 80 bit security, the size of the secret key used in the proposed scheme is reduced by about 75% and the speed of signature generation is accelerated by about 34%.

1.1 Related Works

There are several cryptographic schemes constructed over non-commutative rings. Some encryption schemes using the Braid group have been proposed [1,14]. Polly Cracker has been extended to non-commutative versions [24]. Moreover, Sato and Araki proposed a scheme in the quaternion ring over $\mathbb{Z}/N\mathbb{Z}$ for composite number N [25], which Hashimoto and Sakurai extended to the case of more variables [13]. These schemes constructed over $\mathbb{Z}/N\mathbb{Z}$ were not aimed at resisting quantum attacks, but then Yasuda and Sakurai presented a scheme replacing $\mathbb{Z}/N\mathbb{Z}$ with finite fields [29]. The scheme proposed by Yasuda and Sakurai is essentially equivalent to the proposed NC-Rainbow, in a special case: namely, when the size of each layer is fixed. On the other hand, the proposed NC-Rainbow can choose layers of flexible size. In this paper we discuss the details of secure parameters of the proposed scheme against known attacks and estimate their efficiency in the quaternion ring.

This paper is organized as follows. In §2 we provide a brief overview of Rainbow and its key size. In §3 we describe the known attacks against Rainbow generally. In §4 we present NC-Rainbow, a new variant of Rainbow using non-commutative rings and with the public and secret key sizes of Rainbow. In §5 we discuss the correspondence between the proposed NC-Rainbow and Rainbow, and we analyze the security of the proposed scheme. In §6 we present some of the parameters of NC-Rainbow at a security level of greater than 80 bits, and we compare the secret key sizes and efficiency of NC-Rainbow with those of corresponding Rainbow. Finally, in §7 we provide some concluding remarks.

2 Original Rainbow

Ding and Schmidt proposed a signature scheme called Rainbow, which is a multilayer variant of Unbalanced Oil and Vinegar [8].

First, we define the parameters that determine the layer structure of Rainbow. Let t be the number of layers in Rainbow. Let $v_1, \ldots, v_{t+1}$ be a sequence of $t+1$ positive integers such that

$$0 < v_1 < v_2 < \cdots < v_t < v_{t+1}.$$

For $i = 1, \ldots, t$, the set of indices of the i-th layer in Rainbow is defined by all integers from v_i to v_{i+1}, namely

$$O_i = \{v_i + 1, v_i + 2, \ldots, v_{i+1} - 1, v_{i+1}\}.$$

The number of indices for the i-th layer, O_i is then $v_{i+1} - v_i$, and this is denoted by $o_i = v_{i+1} - v_i$. Note that the smallest integer in O_1 is $v_1 + 1$. We then define $V_1 = \{1, 2, \ldots, v_1\}$, and for $i = 2, 3, \ldots, t+1$, we have

$$V_i = V_1 \cup O_1 \cup O_2 \cup \cdots \cup O_{i-1} = \{1, 2 \ldots, v_i\}.$$

The number of elements in V_i is exactly v_i for $i = 1, 2, \ldots, t+1$. The sets O_i and V_i are used for the indices of the Oil and Vinegar variables in Rainbow, respectively. We define $n = v_{t+1}$, which is the maximum number of variables used in Rainbow.

Next, let K be a finite field of order q. Rainbow consists of t layers of n variables polynomials. For $h = 1, 2, \ldots, t$, the h-th layer of Rainbow deploys the following system of o_h multivariate polynomials:

$$g_k(x_1, \ldots, x_n) = \sum_{i \in O_h, j \in V_h} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V_h,\, i \le j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V_{h+1}} \gamma_i^{(k)} x_i + \eta^{(k)} \qquad (k \in O_h), \tag{1}$$

where $\alpha_{i,j}^{(k)}, \beta_{i,j}^{(k)}, \gamma_i^{(k)}, \eta^{(k)} \in K$. We call the variables x_i $(i \in O_h)$ and x_j $(i \in V_j)$ Oil and Vinegar variables, respectively. The central map of Rainbow is then constructed according to

$$G = (g_{v_1+1}, \ldots, g_n) : K^n \to K^{n-v_1}.$$

Note that a system of o_h equations,

$$g_k(b_1, \ldots, b_{v_h}, x_{v_h+1}, \ldots, x_{v_{h+1}}) = a_k \quad (k \in O_h)$$

becomes o_h linear equations in o_h variables for any $(a_{v_h+1}, \ldots, a_{v_{h+1}}) \in K^{o_h}$ and $(b_1, \ldots, b_{v_h}) \in K^{v_h}$. Therefore, once we know the values of the Oil variables in the h-th layer, we can then compute the values of the Vinegar variables in the $(h+1)$-th layer. This is a trapdoor mechanism in Rainbow.

2.1 Scheme

We describe the key generation, signature generation and verification processes of Rainbow as follows.

Key Generation. A secret key consists of a central map G and two affine transformations $A_1 : K^m \to K^m$ $(m = n - v_1)$, $A_2 : K^n \to K^n$. The public key consists of the field K and the composed map $F = A_1 \circ G \circ A_2 : K^n \to K^m$, which is a system of m quadratic polynomials of n variables over K. We denote the public key by $F = (f_{v_1+1}, \ldots, f_n)^{\mathrm{T}}$ where T denotes a transpose operation. In addition, we call f_k the k-th public polynomial of F for $k = v_1 + 1, \ldots, n$.

Signature Generation. Let $\mathbf{M} \in K^m$ be a message. We compute $\mathbf{A} = A_1^{-1}(\mathbf{M})$, $\mathbf{B} = G^{-1}(\mathbf{A})$ and $\mathbf{C} = A_2^{-1}(\mathbf{B})$ in that order. The signature of the message is $\mathbf{C} \in K^n$. Note that $\mathbf{B} = G^{-1}(\mathbf{A})$ can be easily computed on basis of the above-mentioned property of G.

Verification. If $F(\mathbf{C}) = \mathbf{M}$, the signature is accepted, otherwise it is rejected.

This scheme is denoted as Rainbow$(K;\, v_1, o_1, \ldots, o_t)$, and we call $v_1, o_1, \ldots, o_t$ the parameters of Rainbow.

2.2 Rainbow Key Sizes

We estimate the public and secret key sizes of Rainbow$(K;\, v_1, o_1, \ldots, o_t)$ as follows. Recall that $n = v_1 + o_1 + \cdots + o_t$ and $m = n - v_1$. The public key F is a system of m multivariate polynomials of n variables over K. The secret key consists of central map G, which is a system of m multivariate polynomials of n variables over K, and affine maps $A_1 : K^m \to K^m, A_2 : K^n \to K^n$. Therefore the public and secret key sizes of Rainbow$(K;\, v_1, o_1, \ldots, o_t)$ can be estimated from the number of elements in field K.

Public Key Size

$$\frac{m(n+1)(n+2)}{2} \text{ field elements.}$$

Secret Key Size

$$m(m+1) + n(n+1) + \sum_{h=1}^{t} o_h \left(v_h o_h + \frac{v_h(v_h+1)}{2} + v_{h+1} + 1 \right) \text{ field elements.}$$

3 Attacks against Rainbow

In this section, we summarize the known attacks against Rainbow that have been reported in previous papers, and we analyze the security against each because these are used in our proposed scheme, which is described in the subsequent section. The known relevant attacks against Rainbow are as follows.

(1) Direct attacks [2,28],
(2) UOV attack [16,15],
(3) MinRank attack [12,27,4],
(4) HighRank attack [12,9,20],
(5) Rainbow-Band-Separation (RBS) attack [9,19],
(6) UOV-Reconciliation (UOV-R) attack [9,19].

The direct attacks try to solve a system of equations $F(\mathbf{X}) = \mathbf{M}$ from public key F and (fixed) message $\mathbf{M}$ [2,28]. By contrast, the goal of the other attacks is to find a part of the secret key. In the case of a UOV attack or HighRank attack, for example, the target Rainbow with parameters $v_1, o_1, \ldots, o_t$ is then reduced into a version of Rainbow with simpler parameters such as $v_1, o_1, \ldots, o_{t-1}$ without o_t. We can then break the original Rainbow with lower complexity. To carry out a reduction we need to find (a part of) a direct sum decomposition of vector space K^n,

$$K^n = K^{v_1} \oplus K^{o_1} \oplus \cdots \oplus K^{o_t}, \tag{2}$$

because expressing K^n in an available basis enables returning the public key to the central map. In fact, if we can decompose $K^n = W \oplus K^{o_t}$ for a certain W that has a coarser decomposition than (2) then the security of Rainbow$(K; v_1, o_1, \ldots, o_t)$ can be reduced to that of Rainbow $(K; v_1, o_1, \ldots, o_{t-1})$. There are two methods for finding this decomposition:

(1) Find a simultaneous isotropic subspace of K^n.
Let V be a vector space over K, and let Q_1 be a quadratic form on V. We determine that a subspace W of V is *isotropic* (with respect to Q_1) if

$$v_1, v_2 \in W \Rightarrow Q_1(v_1, v_2) := Q_1(v_1 + v_2) - Q_1(v_1) - Q_1(v_2) = 0.$$

In addition, we assume that V is also equipped with quadratic forms $Q_2, \ldots, Q_m$. We determine that a subspace W of V is *simultaneously isotropic* if W is isotropic with respect to all $Q_1, \ldots, Q_m$.

In Rainbow, m quadratic forms on K^n are defined by the quadratic parts of the public polynomials of F. Note that the subspace K^{o_t} appearing in (2) is a simultaneous isotropic subspace of K^n. If we find a simultaneous isotropic subspace, the basis of K^{o_t} is then obtained and the above attack is feasible. The UOV, UOV-R and RBS attacks are classified as being of this type.

(2) Find a quadratic form with the minimum or second maximum rank.
When the quadratic part of the k-th public polynomial of F in Rainbow is expressed as

$$\sum_{i=1}^{n}\sum_{j=i}^{n} a_{ij}^{(k)} x_i x_j,$$

we associate it with a symmetric matrix $P_k = A + A^{\mathrm{T}}$, where $A = (a_{ij}^{(k)})$. We define $\Omega_F = \mathrm{Span}_K\{P_k \mid k = v_1 + 1, \ldots, n\}$, which is a vector space over K spanned by matrices $P_{v_1+1}, \ldots, P_n$. For example, if we find a matrix of rank $v_2 = v_1 + o_1$ in Ω_F, there is a high probability that the image of this matrix coincides with $K^{v_1} \oplus K^{o_1}$ appearing in (2). Therefore, we obtain the decomposition of $K^n = (K^{v_1} \oplus K^{o_1}) \oplus W'$ for some W' that is a coarser decomposition than (2). The MinRank and HighRank attacks are classified as being of this type.

The details of abovementioned six attacks can be found in the literature [19].

4 Our Proposed Scheme

In this section, we propose NC-Rainbow, a variant of Rainbow that utilizes non-commutative rings. After describing some of the basic properties of non-commutative rings, we explain the construction of the proposed scheme, and estimate its key size.

4.1 Non-commutative Rings

Let R be a finite dimensional non-commutative K-algebra [10], namely, a non-commutative ring satisfying the following condition:

(1) R is a vector space over K with finite dimension.
(2) $\alpha(vw) = (\alpha v)w = v(\alpha w) \quad (\forall \alpha \in K, \forall v, \forall w \in R)$.

In the remainder of this paper, we simply call any R that satisfies the above conditions a non-commutative ring.

Example 1 (Quaternion algebra). Let K' be any quadratic extension field of K. For $b \in K^{\times}$, a non-commutative ring $Q_q(b)$ is defined as followsF

$$\begin{aligned} &\text{(Set)} \quad Q_q(b) = K' \cdot 1 \oplus K' \cdot e, \\ &\text{(Product)} \quad e^2 = b,\ \alpha e = e\bar{\alpha} \quad (\forall \alpha \in K'). \end{aligned}$$

$Q_q(b)$ is a four-dimensional K-space. This is called a quaternion algebra. We write Q_{256} for $Q_{256}(-1)$ which is used later.

The following is a well-known and important property of a non-commutative ring.

Proposition 1. *For a non-commutative ring R, there is $d \in \mathbb{N}$ such that there exists an injective ring homomorphism,*

$$R \hookrightarrow \mathbb{M}(d, K).$$

In particular, using the regular representation we can take $d = r$. Here $\mathbb{M}(d, K)$ is the full matrix ring consisting of $d \times d$ matrices with K entries.

We fix a non-commutative ring R, and r denotes its dimension over K. Then, there exists a K-linear isomorphism,

$$\phi : K^r \xrightarrow{\sim} R. \tag{3}$$

Using the isomorphism ϕ, an element $\alpha \in R$ can be represented by r elements in K.

4.2 Our Construction

In this section, we propose a new Rainbow scheme. The basic idea of the proposed scheme is to change the field K over which Rainbow is constructed into a non-commutative ring R. To avoid notational confusion, we change the parameters used in Rainbow$(K; v_1, o_1, \ldots, o_t)$ and $n = v_1 + o_1 + \cdots + o_t, m = n - v_1$ as follows:

$$t \to s,\ n \to \tilde{n},\ m \to \tilde{m},\ v_i \to \tilde{v}_i,\ o_i \to \tilde{o}_i,\ \ldots$$

Using these parameters, we will construct the proposed Rainbow in the same manner as Rainbow. Let $\tilde{n}, s, \tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{s+1}$ be integers such that

$$0 < \tilde{v}_1 < \tilde{v}_2 < \cdots < \tilde{v}_s < \tilde{v}_{s+1} = \tilde{n}.$$

For $i = 1, \ldots, s$, we set the indices of the Oil and Vinegar variables in the proposed scheme, as

$$\tilde{V}_i = \{1, \ldots, \tilde{v}_i\}, \qquad \tilde{O}_i = \{\tilde{v}_i + 1, \ldots, \tilde{v}_{i+1}\}.$$

Note that the number of elements in $\tilde{V}_i$ and $\tilde{O}_i$ is $\tilde{v}_i$ and $\tilde{o}_i := \tilde{v}_{i+1} - \tilde{v}_i$, respectively. The proposed Rainbow scheme consists of s layers of $\tilde{n}$ variables multivariate polynomials. For the h-th layer of the proposed Rainbow scheme $(h = 1, 2, \ldots, s)$, we deploy the following system of $\tilde{o}_h$ variables polynomials over the non-commutative ring R:

$$\begin{aligned}\tilde{g}_k(x_1, \ldots, x_{\tilde{n}}) = &\sum_{i \in \tilde{O}_h, j \in \tilde{V}_h} (x_i \alpha_{i,j}^{(k)} x_j + x_j \alpha_{i,j}^{(k)} x_i) + \sum_{i,j \in \tilde{V}_h} x_i \beta_{i,j}^{(k)} x_j \\ &+ \sum_{i \in \tilde{V}_{h+1}} (\gamma_i^{(k,1)} x_i + x_i \gamma_i^{(k,2)}) + \eta^{(k)} \quad (k \in \tilde{O}_h). \end{aligned} \tag{4}$$

where $\alpha_{i,j}^{(k)}, \beta_{i,j}^{(k)}, \gamma_i^{(k,1)}, \gamma_i^{(k,2)}, \eta^{(k)} \in R$. Note that there are other possibilities in constructing $\tilde{g}_k$, such as changing $x_i \alpha_{i,j}^{(k)} x_j$ into $\alpha_{i,j}^{(k)} x_i x_j$, due to the non-commutative property of R. We have chosen this construction of (4) because

it has a relatively small number of coefficients, and yet, has the same level of security. We will discuss the efficiency and security of this construction in the subsequent section.

Using equation (4), the central map of the proposed Rainbow scheme is constructed by

$$\tilde{G} = (\tilde{g}_{v_1+1}, \ldots, \tilde{g}_{\tilde{n}}) : R^{\tilde{n}} \to R^{\tilde{m}} \quad (\tilde{m} = \tilde{n} - \tilde{v}_1).$$

The key generation and the signature generation and their verification are described as follows.

Key Generation. A secret key consists of the above central map, $\tilde{G} : R^{\tilde{n}} \to R^{\tilde{m}}$, and two affine transformations, $A_1 : K^m \to K^m$ $(m = r\tilde{m})$, $A_2 : K^n \to K^n$ $(n = r\tilde{n})$. Note that $\tilde{G}$ is a map over the non-commutative ring R. We need to convert $\tilde{G}$ into a map over K using ϕ in (3). The public key is then the composed map $\tilde{F} = A_1 \circ \phi^{-\tilde{m}} \circ \tilde{G} \circ \phi^{\tilde{n}} \circ A_2 : K^n \to K^m$.

Signature Generation. Let $\mathbf{M} \in K^m$ be a message. We compute $\mathbf{A} = (\phi^{\tilde{m}} \circ A_1^{-1})(\mathbf{M})$, $\mathbf{B} = \tilde{G}^{-1}(\mathbf{A})$ and $\mathbf{C} = (A_2^{-1} \circ \phi^{-\tilde{n}})(\mathbf{B})$ in that order. The signature of the message is $\mathbf{C} \in K^n$. Here $\mathbf{B} = \tilde{G}^{-1}(\mathbf{A})$ is computed using the following procedure.

Step 1 Choose a random element $b_1, \ldots, b_{\tilde{v}_1} \in R$.

Step 2 For $h = 1, \ldots, s$, recursively conduct the following operation (called the h-th layer):

$\{\tilde{g}_{\tilde{v}_h+1}, \ldots, \tilde{g}_{\tilde{v}_{h+1}}\}$ is a system of non-commutative polynomials with respect to $x_1, \ldots, x_{\tilde{v}_{h+1}}$. By substituting $x_1 = b_1, \ldots, x_{\tilde{v}_h} = b_{\tilde{v}_h}$ into this system, a new system $\{\bar{g}_{\tilde{v}_h+1}, \ldots, \bar{g}_{\tilde{v}_{h+1}}\}$ of non-commutative polynomials with at most one degree with respect to $x_{\tilde{v}_h+1}, \ldots, x_{\tilde{v}_{h+1}}$ is obtained.

$$\begin{cases} \bar{g}_{\tilde{v}_h+1}(x_{v_h+1}, \ldots, x_{\tilde{v}_{h+1}}) = a_{\tilde{v}_h+1} \\ \qquad\qquad \vdots \\ \bar{g}_{\tilde{v}_{h+1}}(x_{\tilde{v}_h+1}, \ldots, x_{\tilde{v}_{h+1}}) = a_{\tilde{v}_{h+1}} \end{cases} \tag{5}$$

where $\mathbf{A} = (a_k) \in R^{\tilde{m}}$. We compute the solution $b_{\tilde{v}_h+1}, \ldots, b_{\tilde{v}_{h+1}} \in R$ for this system of equations (if there is no solution, return to Step 1.)

Step 3 Set $\mathbf{B} = (b_1, \ldots, b_{\tilde{n}})$.

Verification. If $\tilde{F}(\mathbf{C}) = \mathbf{M}$, the signature is accepted; otherwise it is rejected.

This scheme is denoted by NC-Rainbow(R; $\tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_t$), and we call $\tilde{v}_1, \ldots, \tilde{o}_t$ a parameter of non-commutative Rainbow.

Remark 1. In general, it is difficult to solve the system of non-commutative equations (5) directly. However, if we fix a K-basis of R then this system produces a system of commutative equations that has at most one degree with respect to the coefficients with respect to the basis and thus is easy to solve in general.

4.3 Key Size in NC-Rainbow

We estimate the public and secret key sizes of NC-Rainbow(R; $\tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_t$) as follows. Recall that an element $\alpha \in R$ can be represented by r elements in K using the isomorphism ϕ in (3). The equation (4) appearing in central map $\tilde{G}$ has $2\tilde{v}_h\tilde{o}_h + \tilde{v}_h^2 + 2\tilde{v}_{h+1} + 1$ coefficients, and there are o_h polynomials in the h-th layer, with a total of s layers. Thus, we can estimate the key size of the proposed Rainbow based on the number of elements in field K as described in §2.2. The public and secret key sizes of NC-Rainbow(R; $\tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_t$) are as follows.

Public Key Size

$$\frac{r\tilde{m}(r\tilde{n}+1)(r\tilde{n}+2)}{2} \text{ field elements.}$$

Secret Key Size

$$r\tilde{m}(r\tilde{m}+1) + r\tilde{n}(r\tilde{n}+1) + \sum_{h=1}^{s} r\tilde{o}_h \left(2\tilde{v}_h\tilde{o}_h + \tilde{v}_h^2 + 2\tilde{v}_{h+1} + 1\right) \text{ field elements.}$$

Note that by substituting $n = r\tilde{n},\ m = r(\tilde{n} - \tilde{v}_1)$, the public key size becomes $m(n+1)(n+2)/2$, which is the same as that of the original Rainbow.

5 Security Analysis

In this section we analyze the security of our proposed NC-Rainbow. First we discuss the relation between the proposed NC-Rainbow and the original Rainbow. The security of the proposed NC-Rainbow against known attacks is then analyzed.

5.1 Reducing NC-Rainbow to Rainbow

In what follows we describe the correspondence between the proposed scheme and the original Rainbow. We will show that the public key of the proposed scheme can be represented as a public key of the original Rainbow.

Theorem 1. *Let R be a non-commutative ring of dimension r over K. Let $\tilde{F}$ be a public key of NC-Rainbow(R; $\tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_s$). Then $\tilde{F}$ becomes a public key of Rainbow(K; $r\tilde{v}_1, r\tilde{o}_1, \ldots, r\tilde{o}_s$).*

Proof. It suffices to show that for the central map $\tilde{G}$ of NC-Rainbow(R; $\tilde{v}_1, \ldots, \tilde{o}_s$), $\phi^{-\tilde{m}} \circ \tilde{G} \circ \phi^{\tilde{n}}$ is a central map of Rainbow(K; $r\tilde{v}_1, r\tilde{o}_1, \ldots, r\tilde{o}_s$). Fix an K-basis of the isomorphism ϕ in (3) as

$$e_i = \phi((0, \cdots, 0, \overset{i}{1}, 0, \cdots, 0)) \in R \quad (i = 1, \ldots, r).$$

Denote the decomposition of the non-commutative variables x and x' in R by

$$x = \sum_{j=1}^{r} \bar{x}_j e_j, \quad x' = \sum_{j=1}^{r} \bar{x}'_j e_j.$$

A component $\tilde{g}_k$ of the central map $\tilde{G}$ where k is in the h-th layer $\tilde{O}_h$ is described as a sum of a constant and the following non-commutative monomials:

$$g_{1,\alpha}(x, x') = x\alpha x', \;\; g_{2,\alpha}(x) = x\alpha x,$$
$$h_{1,\alpha}(x) = \alpha x, \;\; h_{2,\alpha}(x) = x\alpha,$$

for some $\alpha \in R$ using the appropriate substitutions $x = x_i$ and $x' = x_j$. For example, suppose that $g_{1,\alpha}$ is a term of $\tilde{g}_k$ under the substitution $x = x_i$. If we write

$$g_{1,\alpha}(x, x') = \sum_{j=1}^{r} f_j^{(\alpha)}(\bar{x}_1, \ldots, \bar{x}_r, \bar{x}'_1, \ldots, \bar{x}'_r)\, e_j,$$

each $f_j^{(\alpha)}(\bar{x}_1, \ldots, \bar{x}_r, \bar{x}'_1, \ldots, \bar{x}'_r)$ is a quadratic homogeneous polynomial. Since

$$\phi^{-1} \circ g_{1,\alpha} \circ \phi^2 = (f_1^{(\alpha)}, \ldots, f_r^{(\alpha)}) : K^{2r} \to K^r,$$

when g_k is expressed using variables $\bar{x}_1, \ldots, \bar{x}_r, \bar{x}'_1, \ldots, \bar{x}'_r$, the part corresponding to $g_{1,\alpha}$ is described as a quadratic homogeneous polynomial. Similarly, in the case of $g_{2,\alpha}(x)$, $\phi^{-1} \circ g_{2,\alpha} \circ \phi : K^r \to K^r$ is described as a quadratic homogeneous polynomial, and if $h = h_{1,\alpha},\ h_{2,\alpha}$, $\phi^{-1} \circ h \circ \phi : K^r \to K^r$ is described as a linear homogeneous polynomial. Moreover, if $f_1, \ldots, f_r$ are defined by

$$\phi^{-1} \circ \tilde{g}_k \circ \phi^{\tilde{n}} = (f_1, \ldots, f_r) : K^n \to K^r,$$

then, by the replacement $x_i = \sum_{l=1}^{r} \overline{x}_{ri-r+l}\, e_l$ $(i = 1, \ldots, \tilde{n})$, we have

$$f_l(\overline{x}_1, \ldots, \overline{x}_n) = \sum_{i \in O_h, j \in V_h} \alpha_{i,j}^{(l)} \overline{x}_i \overline{x}_j + \sum_{i,j \in V_h,\, i \le j} \beta_{i,j}^{(l)} \overline{x}_i \overline{x}_j + \sum_{i \in V_{h+1}} \gamma_i^{(l)} \overline{x}_i + \eta^{(l)} \quad (l = 1, \ldots, r).$$

Here $\alpha_{i,j}^{(l)}, \beta_{i,j}^{(l)}, \gamma_i^{(l)}, \eta^{(l)} \in K$ and

$$V_h = \{1, \ldots, r\tilde{v}_h\}, \quad O_h = \{r\tilde{v}_h + 1, \ldots, r\tilde{v}_{h+1}\}.$$

Thus f_l is expressed in the form of a component with an index in the h-th layer of a central map of Rainbow$(K;\, r\tilde{v}_1, r\tilde{o}_1, \ldots, r\tilde{o}_s)$. This implies that $\phi^{-\tilde{m}} \circ \tilde{G} \circ \phi^{\tilde{n}}$ is a central map of Rainbow$(K;\, r\tilde{v}_1, r\tilde{o}_1, \ldots, r\tilde{o}_s)$. □

This theorem shows that the proposed NC-Rainbow is an additional method for constructing Rainbow. This means that attacks against Rainbow can also be applied to NC-Rainbow, and we can thus evaluate the security provided by the latter scheme against such attacks.

5.2 Security against Known Attacks

UOV Attack. Regard L_2 as the part of a linear transformation of A_2 and place $\mathcal{O}_t = L_2^{-1}(\{0\}^{n-o_t} \times K^{o_t})$ as the subspace of K^n corresponding to K^{o_t} appearing in (2). The UOV attack finds a non-trivial invariant subspace of $W_{12} = W_1 W_2^{-1}$ that is included in $\mathcal{O}_t$ for invertible matrices $W_1, W_2 \in \Omega_F$. The probability that W_{12} has a non-trivial invariant subspace included in $\mathcal{O}_t$ is equal to q^{n-2o_t}. This is obtained by the following lemma.

Lemma 1 ([7] Lemma 3.2.4). *Let $J : K^n \to K^n$ be an invertible linear map such that*

1. *there exist two subspace $\mathcal{O}' \subset \mathcal{V}'$ of K^n where the dimensions of $\mathcal{O}'$ and $\mathcal{V}'$ are o' and v', respectively, and*
2. *$J(\mathcal{O}') \subset \mathcal{V}'$.*

Then the probability that J has a non-trivial invariant subspace in $\mathcal{O}'$ is no less than $q^{o'-v'}$.

This lemma is also available for NC-Rainbow. This means that the complexity is the same as that of the corresponding Rainbow. Thus, we have the following proposition:

Proposition 2. *For $K = GF(2^a)$, NC-Rainbow$(R; \tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_s)$ has a security level of l bits against the UOV attack if*

$$n - 2r\tilde{o}_s \geq l/a + 1, \quad (n = r\tilde{n}).$$

MinRank Attack. In the MinRank attack, we solve MinRank(v_2) for Ω_F. If there is a non-trivial $P \in \Omega_F$ for a $v \in K^n$ such that $Pv = 0$, there is high probability that P is a solution for MinRank(v_2). For $v \in K^n$, the probability that a non-trivial $P \in \Omega_F$ exists such that $Pv = 0$ is roughly q^{-v_2}. This is also true for NC-Rainbow. Therefore from [12], we have the following proposition:

Proposition 3. *Let $K = GF(2^a)$ and assume that $n \geq m \geq 10$. Then NC-Rainbow$(R; \tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_s)$ has a security level of l bits against the MinRank attack if*

$$r\tilde{v}_2 = r(\tilde{v}_1 + \tilde{o}_1) \geq l/a.$$

HighRank Attack. In the HighRank attack, we have an element $W \in \Omega_F$ such that $\mathrm{rank}(W) = v_t$. For any $W \in \Omega_F$, the probability that its rank is equal to v_t is q^{-o_t}. This is also true for NC-Rainbow. Therefore, from [12], we have the following proposition:

Proposition 4. *Let $K = GF(2^a)$ and assume that $n \geq m \geq 10$. Then NC-Rainbow$(R; \tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_s)$ has a security level of l bits against the HighRank attack if*

$$r\tilde{o}_s \geq l/a.$$

Direct Attacks and Others. From Theorem 1, the public key of NC-Rainbow is exactly equal to that of the corresponding Rainbow. Therefore, the complexity against the direct attacks is estimated to be the same for NC-Rainbow as for the original Rainbow corresponding to it. Similarly, the complexities against the RBS and UOV-R attacks are estimated to be the same for NC-Rainbow as for the corresponding Rainbow.

The complexities of the direct, RSB and UOV-R attacks were discussed by Petzoldt et al. [19], and we follow their data regarding the complexities of these attacks. In particular, if $\tilde{v}_1 \geq \tilde{o}_s$, the complexities of the direct and UOV-R attacks are equivalent.

6 Key Size in Our NC-Rainbow

In this section, we explain why the secret key size of the proposed scheme is reduced, and we then present secure parameters for a security level of greater than 80 bits.

6.1 Reason for Secret Key Size Reduction

First, we describe the theoretical reason that the secret key size in NC-Rainbow is reduced. If we fix a K-basis for R, based on the right regular action,

$$R \ni a \mapsto ar \in R,$$

the embedding $R \hookrightarrow \mathbb{M}(r, K)$ is obtained (Proposition 1). In general, for $d \in \mathbb{N}$, we have the following embedding

$$\phi : \mathbb{M}(d, R) \hookrightarrow \mathbb{M}(dr, K).$$

Table 1. Security level against attacks on NC-Rainbow(Q_{256}; $\tilde{v}_1, \tilde{o}_1, \tilde{o}_2$)

Attacks	$(5,4,4)$	$(7,5,5)$	$(9,6,6)$
Direct, UOV-R, RBS (bits)	83	96	107
UOV (bits)	152	216	280
MinRank (bits)	288	384	480
HighRank (bits)	128	160	192
Security level	83	96	107

Table 2. Secret key sizes of NC-Rainbow and Rainbow

Proposed NC-Rainbow($Q_{256}; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2$)	$(5,4,4)$	$(7,5,5)$	$(9,6,6)$
Secret key size (kB)	8.0	15.1	25.5
	⇓ corresponding to ⇓		
Rainbow($GF(256); 4\tilde{v}_1, 4\tilde{o_1}, 4\tilde{o_2}$)	$(20,16,16)$	$(28,20,20)$	$(36,24,24)$
Secret key size (kB)	33.6	70.7	128.2
Ratio	23.9%	21.5%	19.9%

Here, an element of $\mathbb{M}(d, R)$ can be described using d^2r field elements, while an element of $\mathbb{M}(dr, K)$ requires using d^2r^2 field elements. In NC-Rainbow, a matrix is expressed by ϕ. Since a secret key is described through matrices its size is reduced. This is one reason for a reduction in the key size. Another reason is that the number of equations in NC-Rainbow is $1/r$ times that in the original Rainbow.

6.2 Secure Parameters and Their Key Size

Based on the security analysis in the last section, we try to present secure parameters and their length for NC-Rainbow(R; $\tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_s$). We adopt the parameters of Petzoldt et al. [19] for estimating the security against the direct and RBS attacks. Next, we discuss the other types of attacks. From Propositions 2, 3 and 4, the following criteria are used for l-bit security against these attacks: Let a be the bit length of q and r the dimension of R. For NC-Rainbow(R; $\tilde{v}_1, \tilde{o}_1, \ldots, \tilde{o}_s$), we have $n = r\tilde{n},\ m = r(\tilde{n} - \tilde{v}_1)$ and we assume that $n \geq m \geq 10$.

1. UOV attack $\quad n - 2r\tilde{o}_s \geq l/a + 1.$
2. MinRank attack $\quad r(\tilde{v}_1 + \tilde{o}_1) \geq l/a.$
3. HighRank attack $\quad r\tilde{o}_s \geq l/a.$
4. UOV-R attack $\quad \tilde{v}_1 \geq \tilde{o}_s \quad (+\, l\text{-bits security against direct attack}).$

Table 1 presents exemplary parameters of NC-Rainbow(Q_{256}; $\tilde{v}_1, \tilde{o}_1, \tilde{o}_2$) over the quaternion Q_{256} (see §4.1 for the definition) over $GF(256)$, as well as the complexity against each attack. The examples in the table have a security level of greater than 80 bits. Table 2 shows the ratio of between the secret key sizes of Rainbow and NC-Rainbow for these examples. The secret key size of the proposed NC-Rainbow is about 75% shorter than that of the corresponding Rainbow.

6.3 Efficiency Comparison

Table 3 compares the efficiencies of the proposed NC-Rainbow and the corresponding Rainbow. The proposed NC-Rainbow is constructed over the quaternion Q_{256} which can be expressed by a subring of 2×2 matrices of the finite field $GF(256)$. We compare the efficiency with the corresponding Rainbow over the finite field $GF(256)$ of the same security level in Table 2. Therefore we estimate the number of multiplication of $GF(256)$ for an efficiency comparison. The numbers in Table 3 represent the numbers of multiplications in the signature generation. In the signature generation of our proposed scheme, we need to solve, $2\tilde{o}_i$ linear equations over $GF(256)$ for the i-th layer, Conversely, the corresponding Rainbow requires $4\tilde{o}_i$ linear equations over $GF(256)$ for the same i-th layer. Therefore the signature generation of our proposed scheme is about 34% faster than that of the corresponding Rainbow.

Table 3. Efficiency comparison of the proposed NC-Rainbow with the corresponding Rainbow (in terms of the number of multiplications in $GF(2^8)$)

NC-Rainbow$(Q_{256}; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2)$	$(5,4,4)$	$(7,5,5)$	$(9,6,6)$
Proposed NC-Rainbow	46452	97594	176624
Corresponding Rainbow	73236	153314	276832
Ratio	63.4%	63.7%	63.8%

7 Concluding Remarks

We proposed a new construction of Rainbow, called NC-Rainbow, which utilizes non-commutative rings. A non-commutative ring has a canonical embedding, which yields a compact representation of elements, and thus the proposed NC-Rainbow is able to reduce the secret key size in comparison with Rainbow. We also proved that for any public key of NC-Rainbow there exists a corresponding original Rainbow whose public key is the same, and thus, we can analyze the security of NC-Rainbow by applying the known attacks against Rainbow. Finally, we presented several secure parameters of the proposed scheme at a security level of greater than 80 bits. NC-Rainbow reduces the secret key size by about 75% and improves the efficiency of signature generation by about 34%, as compared with Rainbow at the same 80 bit security level.

In this paper, we mainly treated quaternion algebras as non-commutative rings for our proposed NC-Rainbow. Of course, there are many other non-commutative rings such as group rings, simple rings, etc. If we choose different non-commutative rings, the efficiency of NC-Rainbow associated to them also changes. We need further research to investigate the efficiency of NC-Rainbow using different non-commutative rings.

Acknowledgements. This work was partially supported by the Japan Science and Technology Agency (JST) Strategic Japanese-Indian Cooperative Programme for Multidisciplinary Research Fields, which aims to combine Information and Communications Technology with Other Fields. The first author is supported by the JST A-step feasibility study program, No. AS231Z03613A. The authors would like to thank Jintai Ding and the anonymous reviewers for their insightful comments on the draft manuscript.

References

1. Anshel, I., Anshel, M., Goldfeld, D.: An Algebraic Method for Public-Key Cryptography. Math. Res. Lett. 6(3-4), 287–291 (1999)
2. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer, Heidelberg (2009)
3. Berger, T.P., Cayrel, P.-L., Gaborit, P., Otmani, A.: Reducing Key Length of the McEliece Cryptosystem. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 77–97. Springer, Heidelberg (2009)

4. Billet, O., Gilbert, H.: Cryptanalysis of Rainbow. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 336–347. Springer, Heidelberg (2006)
5. Boneh, D., Durfee, G.: Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. IEEE Trans. Inform. Theory 46(4), 1339–1349 (2000)
6. Chen, A.I.-T., Chen, M.-S., Chen, T.-R., Cheng, C.-M., Ding, J., Kuo, E.L.-H., Lee, F.Y.-S., Yang, B.Y.: SSE Implementation of Multivariate PKCs on Modern x86 CPUs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 33–48. Springer, Heidelberg (2009)
7. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate Public Key Cryptosystems. In: Advances in Information Security, vol. 25. Springer, Heidelberg (2006)
8. Ding, J., Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
9. Ding, J., Yang, B.-Y., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M.: New Differential-Algebraic Attacks and Reparametrization of Rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 242–257. Springer, Heidelberg (2008)
10. Farb, B., Dennis, K.: Noncommutative Algebra. Graduate Texts in Mathematics. Springer, Heidelberg (1993)
11. Galbraith, S.D., Ruprai, R.S.: Using Equivalence Classes to Accelerate Solving the Discrete Logarithm Problem in a Short Interval. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 368–383. Springer, Heidelberg (2010)
12. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM Cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
13. Hashimoto, Y., Sakurai, K.: On Construction of Signature Schemes based on Birational Permutations over Noncommutative Ringsh. In: Proceedings of the 1st International Conference on Symbolic Computation and Cryptography (SCC 2008), pp. 218–227 (2008)
14. Ko, K.H., Lee, S., Cheon, J.J.H., Han, J.H., Kang, J.S., Park, C.: New Public-Key Cryptosystems Using Braid Groups. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 166–183. Springer, Heidelberg (2000)
15. Kipnis, A., Patarin, L., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
16. Kipnis, A., Shamir, A.: Cryptanalysis of the Oil and Vinegar Signature Scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
17. van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. Journal of Cryptology 12, 1–28 (1999)
18. Petzoldt, A., Bulygin, S., Buchmann, J.: A Multivariate Signature Scheme with a Partially Cyclic Public Key. In: Proceedings of the Second International Conference on Symbolic Computation and Cryptography (SCC 2010), pp. 229–235 (2010)
19. Petzoldt, A., Bulygin, S., Buchmann, J.: Selecting Parameters for the Rainbow Signature Scheme. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 218–240. Springer, Heidelberg (2010)
20. Petzoldt, A., Bulygin, S., Buchmann, J.: CyclicRainbow - A Multivariate Signature Scheme with a Partially Cyclic Public Key Based on Rainbow. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 33–48. Springer, Heidelberg (2010)

21. Petzoldt, A., Bulygin, S., Buchmann, J.: Linear Recurring Sequences for the UOV Key Generation. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 335–350. Springer, Heidelberg (2011)
22. Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small Public Keys and Fast Verification for Multivariate Quadratic Public Key Systems. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 475–490. Springer, Heidelberg (2011)
23. Pollard, J.M.: Monte Carlo Methods for Index Computation mod p. Mathematics of Computation 143(32), 918–924 (1978)
24. Rai, T.S.: Infinite Gröbner Bases and Noncommutative Polly Cracker Cryptosystems. PhD Thesis, Virginia Polytechnique Institute and State Univ. (2004)
25. Satoh, T., Araki, K.: On Construction of Signature Scheme over a Certain Noncommutative Ringh. IEICE Trans. Fundamentals E80-A, 702–709 (1997)
26. Wiener, M.J.: Cryptanalysis of Short RSA Secret Exponents. IEEE Trans. Inform. Theory 36(3), 553–558 (1990)
27. Yang, B.-Y., Chen, J.-M.: Building Secure Tame like Multivariate Public-Key Cryptosystems: The new TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)
28. Yang, B.-Y., Chen, J.-M.: All in the XL Family, Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
29. Yasuda, T., Sakurai, K.: A Security Analysis of Uniformly-Layered Rainbow Revisiting Sato-Araki's Non-commutative Approach to Ong-Schnorr-Shamir Signature Towards PostQuantum Paradigm. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 275–294. Springer, Heidelberg (2011)

A Duality in Space Usage between Left-to-Right and Right-to-Left Exponentiation

Colin D. Walter

Information Security Group, Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom
Colin.Walter@rhul.ac.uk

Abstract. Most exponentiation algorithms are categorised as being left-to-right or right-to-left because of the order in which they use the digits of the exponent. There is clear value in having a canonical way of transforming an algorithm in one direction into an algorithm in the opposite direction: it may lead to new algorithms, different implementations of existing algorithms, improved side-channel resistance, greater insights. There is already an historic duality between left-to-right and right-to-left exponentiation algorithms which shows they take essentially the same time, but it does not treat the space issues that are always so critical in resource constrained embedded crypto-systems. To address this, here is presented a canonical duality which preserves both time and space. As an example, this is applied to derive a new, fast yet compact, left-to-right algorithm which makes optimal use of recently developed composite elliptic curve operations.

Keywords: Scalar multiplication, multi-base representation, addition chain, division chain, dual chain, exponentiation, elliptic curve cryptography.

1 Introduction

Exponentiation is the highest level arithmetic operation in all the most popular public key crypto-systems, and in Diffie-Hellman, RSA and ECC in particular. There are a number of different algorithms for performing exponentiation [8,7] which have various properties that allow some control over their time efficiency, their use of space resources and their susceptibility to side channel analysis.

The ability to choose between processing an exponent from left to right or from right to left enables implementers to improve side channel resistance (e.g. by avoiding pre-computed tables [13]) or to make use of more efficient composite group operations such as double-and-add, triple-and-add and quintuple-and-add elliptic curve operations [6,10,9]. The direction of treating the exponent bytes may also be determined by the order in which those bytes become available.

These reasons make it of interest to find a canonical way of restructuring an exponentiation algorithm so that it can process the exponent in the opposite direction. An example of what we would like to do in practice is given

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 84–97, 2012.

by comparing the usual left-to-right m-ary algorithm due to Brauer [3] with Yao's right-to-left method [14]. These use the same time and space resources. Moreover, it is clear how to extend both to sliding window versions that make the same use of resources. In general, it would be useful to be able to take any exponentiation algorithm processing the bits in one direction and deduce immediately a corresponding algorithm for processing the bits in the opposite direction. Knuth in his well-known *Semi-numerical Algorithms* [8] describes the *transposition method*, [2] §5, which enables one to reverse the order of processing the exponent and compute the required power using the same time. With care, the same number of squarings and non-squarings occur in the two directions. However, this method does not show how to preserve the space requirements, nor does it provide a canonical re-ordering. Yet the preservation of space usage is of critical importance to achieve when memory is limited, as on a smart card or an embedded cryptographic device, as well as on SSL servers with systolic arrays for processing many exponentiations in parallel. Nevertheless, these space issues do not seem to have been treated satisfactorily in the literature.

The aim of the present work is too provide a canonical duality between the two directions which not only preserves the time usage of an exponentiation algorithm but also makes identical use of memory. This is done by starting with an addition chain which is annotated with the register locations for the inputs and output of each operation. A careful restriction on the allowable operations makes the chain reversible, so that the use of space is clearly the same in both directions. Some additional conditions are required to ensure that the numbers of squaring and non-squaring operations are also the same for the addition chain and its dual. The restriction on allowed operations is essentially just a requirement on the way the chain is presented, and so does not confine the applicability of the method. The additional conditions are natural ones in an efficient system and so are normally satisfied in a practical environment. The main novel contributions here are the establishment of this correct set of allowable operations to make duality possible, identification of the right conditions to preserve the time cost of an exponentiation when the dual is applied, and a proof of this property.

Application of the duality process shows that Brauer's m-ary method has Yao's method as its dual, and *vice versa*. Also, application to Walter's division chain method [11] yields a new compact left-to-right algorithm which can take maximal advantage of recently developed composite elliptic curve operations [6,10,9] because the recoding of the exponent can be tailored to the different relative costs of any desired combinations of squaring and non-squaring operations on the underlying group, namely the elliptic curve in this case.

Finally, having established that the main space and time requirements are the same for an exponentiation algorithm and its dual, there are some secondary space issues to tidy up. When the duality is applied to an addition chain derived from a recoding of the exponent, extra space may be required to store the complete recoding when processed in one direction, but for the other direction the recoding may be generated on-the-fly. One may also require the initial inputs

(the base and/or the exponent) to remain at the end of the exponentiation. This may happen in one direction, whereas they may be overwritten in the other.

2 Notation and Addition Chains

The duality defined here applies to exponentiation schemes which are defined in terms of *addition* or *addition-subtraction chains* [8]. Most exponentiation algorithms first perform a re-coding of the exponent D, and then convert the re-coding into an addition chain which is applied to an element M of some group G to yield the element $C = M^D$. With cryptographic applications in mind, M will be called the *plaintext*, D the *(secret) key*, and C the *ciphertext*. G might be the group of points on an elliptic curve. It will be written multiplicatively so that the operation of interest is $C \leftarrow M^D$, which is reasonably called an *exponentiation*.

In order to obtain a good measure of the computational time for exponentiating, we will assume there are two (probably distinct) algorithms for performing the group operation. The first computes M^2 for any $M \in G$ and is called a *squaring*. When the two arguments of the group operation are known to be identical this algorithm will be used. The other algorithm computes $M_1 \times M_2$ for any $M_1, M_2 \in G$ and is called a (non-squaring) *multiplication*. This will be used whenever it is not possible to guarantee that $M_1 = M_2$. Normally $M_1 \neq M_2$ when this algorithm is applied, but it is possible that $M_1 = M_2$ could occur by chance. However, the same computational cost will be assumed for all applications of it. Lastly, there may also be a unary operation for computing the *inverse* M^{-1} of M, or, more generally, several unary operations $M \to M^s, s \in S$, for a small subset $S \subset \mathbb{Z}$ of integers. This enables us to deal with a Frobenius map as well as inversion. It may be convenient to include squaring in this category. The following definition picks up these distinctions:

Definition 1
i) An addition chain *of* length n *for* D *is a sequence* $D_0, D_1, D_2, \ldots, D_n$ *of integers such that*

a) $D_0 = 1$ *and* $D_n = D$*;*
b) for all $k, 0<k\leq n$*, either there are* $i, j < k, i \neq j$*, such that* $D_i + D_j = D_k$ *or there is an* $i < k$ *such that* $2D_i = D_k$*.*

ii) A (generalised) addition-subtraction chain *of* length n *for* D *is a sequence* $D_0, D_1, D_2, \ldots, D_n$ *of integers such that*

a) $D_0 = 1$ *and* $D_n = D$*;*
b) for all $k, 0<k\leq n$*, either there are* $i, j < k, i \neq j$*, such that* $D_i + D_j = D_k$ *or there are* $i < k$ *and* $s \in S$ *such that* $sD_i = D_k$*.*

These translate into exponentiation schemes for D in the obvious way. The kth step in obtaining $M^D \in G$ is to compute $M^{D_k} = M^{D_i + D_j} = M^{D_i} \times M^{D_j}$ or $M^{D_k} = M^{sD_i} = (M^{D_i})^s$.

Memory locations for holding elements of G will, for convenience, be called *registers* and denoted $R_i, i \in I$, for some small index set I. i (or R_i) will be

called a *location* of $g \in G$ if R_i stores the value of g. In practice, R_i could be any form of memory, perhaps different for each i so that the cost of reading from or writing to R_i may very well depend on the value of i. Such costs generally result in minor differences in execution times between an algorithm and its dual.

In general, for $i, j, k \in I$ the *mu*ltiplicative operation which writes the product of the contents of R_i and R_j into R_k is denoted μ_{ijk}, and the powering operation writing the sth power of the content of R_i into R_k is denoted $\iota_{ik}^{(s)}$ (choosing "ι" for *inverse* because often $s = -1$). It is clear that, once a location for each D_k in an addition or addition-subtraction chain is known, then the chain can be expressed as a sequence of operations of type μ_{ijk} or $\iota_{ik}^{(s)}$. However, to define the dual chain, only the following restricted sets of operators are allowed:

Definition 2. *For $i, j \in I$ with $i \neq j$ and $s \in S$, six sets of operators are defined:*
i) Copying from R_i to R_j is denoted γ_{ij}.
ii) Copying from R_i to R_j combined with initialising R_i to the group identity 1_G is denoted $\gamma_{ij}^{(0)}$.
iii) The multiplicative operation which writes the product of the contents of R_i and R_j into R_j is denoted μ_{ij}.
iv) The multiplicative operation which writes the product of R_i and R_j into R_j and initialises R_i to 1_G is denoted $\mu_{ij}^{(0)}$.
v) The operation which raises the contents of R_i to the power s is denoted $\iota_i^{(s)}$.
vi) The operation swapping the contents of registers R_i and R_j is denoted σ_{ij}.
A location-aware chain *is a finite sequence of such operations.* □

Location-aware chains will also be called *space-aware* chains, especially where the overall space usage rather than individual data movements are of concern.

Any μ_{ijk} or $\iota_{ij}^{(s)}$ can be expressed using a sequence of either one or two of the above operations with no increase in the number or type of multiplicative operations. For example, if $i \neq k \neq j$ then R_j can be first copied to R_k using γ_{jk} and then μ_{ik} completes the process of computing μ_{ijk}. Similarly, any squaring μ_{iik} can be expressed by first using the copy γ_{ik} if $i \neq k$ and then the powering operation $\iota_k^{(2)}$, thereby making all squaring explicit. If required at execution time, the two operations from such splittings can always be recombined into one when deciding the code to execute. Also at execution time many of the initialisations to 1_G in $\gamma_{ij}^{(0)}$ and $\mu_{ij}^{(0)}$ might be skipped as they are mostly redundant.

The swapping operation enables it to be made explicit when data is moved around. It is included for completeness as it may be needed in implementations to put data in a particular location without loosing the data which is already in that location. Later conditions require this (for the *symmetric* property), but data must also be moved around if only certain locations (such as actual registers) can be used for the I/O of an operation. However, from here onwards, and without loss of generality, *swapping* will be ignored in any proofs since it is irrelevant to them and simply complicates the description of where data is.

3 The Dual of a Location-Aware Chain

The operations in Defn. 2 can be represented using matrices, indexed by I. For example, if $A = (a_{st})$ were the matrix for $\mu_{ij}, i \neq j$, then $a_{ss} = 1$ for $s \in I$, $a_{ji} = 1$, and $a_{st} = 0$ otherwise. It is the identity matrix except for an extra non-zero entry at (j, i). This acts from the left on a column vector containing the exponents of the powers of the input M which are in each register, adding the values with indices i and j into the location with index j. In other words, the matrix performs the same addition as an element of an addition chain.

For a device with two memory locations, i.e. $|I| = 2$, matrix examples of each class are, respectively,

$$\gamma_{12} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \gamma_{12}^{(0)} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \mu_{21} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mu_{21}^{(0)} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix},$$
$$\iota_1^{(s)} = \begin{bmatrix} s & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad \sigma_{12} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

This view enables the *transpose* of each operator to be defined to coincide with the transpose of its matrix:

Definition 3. *The* transposes *of the operators in Definition 2 are as follows:* $\gamma_{ij}^{\mathsf{T}} = \mu_{ji}^{(0)}$, $\gamma_{ij}^{(0)\mathsf{T}} = \gamma_{ji}^{(0)}$, $\mu_{ij}^{\mathsf{T}} = \mu_{ji}$, $\mu_{ij}^{(0)\mathsf{T}} = \gamma_{ji}$, $\iota_i^{(s)\mathsf{T}} = \iota_i^{(s)}$ *and* $\sigma_{ij}^{\mathsf{T}} = \sigma_{ij}$.

Clearly the transpose operator $^{\mathsf{T}}$ is a bijection of order two on the set of operations listed in Definition 2. In greater detail, it is the identity on elements listed in parts (v) and (vi), a bijection on the subsets of parts (ii) and (iii), and a bijection between the elements of parts (i) and (iv). Hence the transpose of a list of such operations will also be a list of such operations, so that the following concept of a *dual chain* is well-defined:

Definition 4. *The* dual *of a location-aware chain* $\rho = (\rho_1, \rho_2, \rho_3, \ldots, \rho_n)$ *is the location-aware chain* $\rho^{\mathsf{T}} = (\rho_n^{\mathsf{T}}, \ldots, \rho_3^{\mathsf{T}}, \rho_2^{\mathsf{T}}, \rho_1^{\mathsf{T}})$.

The foregoing observations imply that the number and type of the powering operations, such as squarings and inversions, is the same for a chain and its dual. Also, the number of multiplications without initialisation, the number of copyings with initialisation, and the number of swappings are all preserved under application of the dual map. However, the number of multiplications with initialisation and the number of copyings without initialisation are interchanged by the dual. Additional conditions are required to make these numbers equal so that the cost of a space aware chain, in terms of the counts of each type of operation, is unchanged when the dual is taken.

Before tackling these conditions, let us reflect on the choice of operations in Definition 2. The general operations $\mu_{ijk}, i \neq k \neq j$, were omitted because their transposes are too complicated for a sensible definition of a dual chain. However, the remaining cases of multiplicative operations, namely $\mu_{ij}, i \neq j$, are not powerful enough to enable all the required operations to be done.

As a result, the copying operations γ_{ij} need to be included. The need for closure under transpose results in the inclusion of multiplications with initialisation. Lastly, the copying with initialisation arises naturally from the conditions in §4.

3.1 Example

In this example, the notation is illustrated by computing M^{15} using two registers, and starting with the addition chain $(1, 2, 3, 6, 12, 15)$. The construction of the chain under Defn. 1 is usually given explicitly in the form

$$1+1=2,\ 1+2=3,\ 3+3=6,\ 6+6=12,\ 12+3=15.$$

but the three doublings can be exhibited by writing it as

$$2{\times}1=2,\ 1+2=3,\ 2{\times}3=6,\ 2{\times}6=12,\ 12+3=15.$$

The corresponding computation with M is

$$(M^1)^2=M^2;\ M^1{\times}M^2=M^3;\ (M^3)^2=M^6;\ (M^6)^2=M^{12};\ M^{12}{\times}M^3=M^{15}$$

which contains 3 squarings and 2 multiplications.

A minimum of 2 storage locations is required for this, say R_1 and R_2. Suppose that only R_1 may be used for input and output. So it is assumed to hold M after initialisation, and should contain the final value M^{15} at the end of the calculation. Using a vector to give the values in the registers, the computation starts with $(M, \perp)$ in (R_1, R_2) where $\perp$ denotes an undefined or unknown value. As M is still needed after it is squared, it must first be copied: γ_{12} yields values (M, M). Then application of $\iota_2^{(2)}$ creates (M, M^2) and $\mu_{21}^{(0)}$ yields $(M^3, 1_G)$. Here the 1_G is created by the superscript $^{(0)}$, and used to overwrite the M^2 as it is no longer required. This is a feature of the chains of interest that is introduced in the next section in order to obtain a dual of equal computational effort. Of course, the computationally unnecessary, and essentially free, initialisation to 1_G would probably be skipped in practice. Using γ_{12} to create (M^3, M^3) means that M^3 is not lost when the next squaring, $\iota_2^{(2)}$ generates (M^3, M^6). Repeating $\iota_2^{(2)}$ produces (M^3, M^{12}) so that the final multiplication $\mu_{21}^{(0)}$ achieves $(M^{15}, 1_G)$. This has the desired power M^{15} in the desired location R_1 at the end of the calculation. It has also eliminated the unwanted data from R_2, which is again a requirement described in the next section for the chains of interest. Summarising, the sequence of operations and register contents is thus

$$\binom{M}{\perp} \xrightarrow{\gamma_{12}} \binom{M}{M} \xrightarrow{\iota_2^{(2)}} \binom{M}{M^2} \xrightarrow{\mu_{21}^{(0)}} \binom{M^3}{1_G} \xrightarrow{\gamma_{12}} \binom{M^3}{M^3} \xrightarrow{\iota_2^{(2)}} \binom{M^3}{M^6} \xrightarrow{\iota_2^{(2)}} \binom{M^3}{M^{12}} \xrightarrow{\mu_{21}^{(0)}} \binom{M^{15}}{1_G}$$

The transposes of the operations $\gamma_{12}, \iota_2^{(2)}, \mu_{21}^{(0)}, \gamma_{12}, \iota_2^{(2)}, \iota_2^{(2)}, \mu_{21}^{(0)}$ are, in order, $\mu_{21}^{(0)}, \iota_2^{(2)}, \gamma_{12}, \mu_{21}^{(0)}, \iota_2^{(2)}, \iota_2^{(2)}, \gamma_{12}$. Reversing the order yields the dual chain, which acts on the registers thus:

$$\binom{M}{\perp} \xrightarrow{\gamma_{12}} \binom{M}{M} \xrightarrow{\iota_2^{(2)}} \binom{M}{M^2} \xrightarrow{\iota_2^{(2)}} \binom{M}{M^4} \xrightarrow{\mu_{21}^{(0)}} \binom{M^5}{1_G} \xrightarrow{\gamma_{12}} \binom{M^5}{M^5} \xrightarrow{\iota_2^{(2)}} \binom{M^5}{M^{10}} \xrightarrow{\mu_{21}^{(0)}} \binom{M^{15}}{1_G}$$

It corresponds to the different addition chain $(1, 2, 4, 5, 10, 15)$. In fact, at a higher level, the dual of the computation $M \to M^3 \to M^{3\times5}$ is $M \to M^5 \to M^{3\times5}$.

4 Preserving the Number of Multiplications

A standard measure of the time taken by an exponentiation is given by the following *cost* associated with the underlying addition chain. Once the execution time for each type of operation is known, the corresponding weighted sum of the entries in the cost tuple will yield the total time for exponentiation.

Definition 5. *The* cost *of a location-aware chain is the tuple consisting of the numbers of each type of operation in the chain, as classified in Definition 2 and refined to separate the counts of the unary operations in part (v) according to the value of $s \in S$.*

Thus, in particular, the cost of a chain yields separately the numbers of copyings, non-squaring multiplications, squarings and inversions, the first two being divided into two parts according to whether the operation includes an initialisation to 1_G or not. (Finer time measurements may be required [1].) In order to preserve cost when taking the dual of a chain, some extra conditions are required:

Definition 6. *A location-aware chain is said to be* normalised *if it satisfies the following criteria:*
i) There is a prescribed subset of registers, indexed by $I_{IO} \subseteq I$, say, which is used for I/O[1]*.*
ii) The inputs to every operation and the final value in any output register must be defined, i.e. no operation output or final output depends on the initial value of any non-input register.
iii) The initial value of an input register and the output from every operation in the chain must be used, i.e. every operation output is the input to a subsequent multiplicative operation or is the final value in an output register.
iv) 1_G is never explicitly the input to any operation nor explicitly the final value of an output register.
v) If an operation involving two registers does not include an initialisation to 1_G then the value remaining in the non-result register must be used by a subsequent multiplicative operation or be the final value in an output register.

The conditions (*ii*)–(*v*) actually specify which registers are for input and output:

- Registers R_J with $J \in I_{IO}$ will have their initial values used by the chain and their final values must be defined and not explicitly set to 1_G.
- Initial values in registers $R_J, J \in I \setminus I_{IO}$, must not be used, and final values in these registers must be removed by initialising them to 1_G.

Thus all the I/O registers will both import values and export values, but none of the non-I/O registers will either import values or export values. Adding copying operations, with initialisation if necessary, at the end of a chain enables any outputs of the chain to be via the same set of registers as is used for inputs.

[1] All that is needed is to have the same *number* of input registers as output registers rather than the same subset for both. The restriction here is reasonable for hardware.

So this condition mainly imposes a requirement for there to be the same number of outputs as inputs. There need not be just one input – the chain could perform a multi-exponentiation.

Part (iii) means there are no redundant operations. This can be achieved from any space-aware chain simply by deleting operations whose values are not used, i.e. those operations whose output is neither an input to a subsequent operation nor an output of the chain. Although redundant, it is allowed to have registers which do not figure in any operations.

Property (iv) means, for example, that neither of the inputs to any multiplication has been set to 1_G as a result of a previous operation with an initialisation of one of the two named registers. Similarly, the input to a copy or powering operation should not have been set to 1_G by a preceding operation. This does not impose any real restriction on allowable chains. The unary operations with 1_G as an input have 1_G as an output and so can be deleted from the chain without affecting the final output(s). A multiplication with 1_G as an input has an output equal to the other input and so it can be replaced by a copy or deleted entirely according to whether the output is to the register that contains 1_G or not. A copy of 1_G can be removed, and an initialisation attached instead to the previous operation that used value in the register that needs to be set to 1_G.

Condition (v) is easily achieved in any chain by modifying every operation to include an initialisation whenever it makes no difference to the computations performed or the values exported. For any operation involving two registers but with no initialisation, it means that the values in both the named registers will be used by subsequent operations or exported. Unary operations, i.e. powering operations, do not have registers affected by this rule.

Finally, the definition really assumes there are no swapping operations involved. If there is any swapping, then, in the obvious way, the old and new locations of the value in a register need to be taken into account when deciding whether a value is used or exported or has been initialised to 1_G or etc.

The example in section §3.1 is of a normalised chain, as is easily checked. The I/O subset of $I = \{1, 2\}$ is $I_{IO} = \{1\}$. It is clear that all non-trivial intermediate register values are used; the unused intermediate values have all been deliberately set to 1_G and are eventually overwritten. The dual chain is also clearly a normalised chain. Both chains in this example have the same costs: there are three squarings, two copyings, and two multiplications with initialisations.

Theorem 1. *The cost of a normalised location-aware chain is unchanged by taking the dual.*

Proof. For simplicity, and without loss of generality, assume there are no unary operations (i.e. those covered by Defn. 2(v)) and no swapping operations. It has already been observed that the numbers of such operations are not changed by taking the dual, nor are the numbers of copyings with initialisation and multiplications without initialisation.

The proof works by counting the number of instances of 1_G or $\perp$ (undefined) occurring in registers, and equating this 1) to the number of operations that create them and 2) to the number of operations that destroy them. So let γ

be the number of copyings without initialisation, γ_I the number of copyings with initialisation to 1_G and μ_I the number of multiplications with initialisation to 1_G. It will be shown that $\gamma = \mu_I$, from which the theorem follows almost immediately. (In the example of §3.1, there are 2 instances of 1_G, 1 of $\bot$, and $\gamma_I = 0$, $\gamma = 2$, $\mu_I = 2$. Equating the numbers yields $1+\mu_I+\gamma_I = 3 = 1+\gamma+\gamma_I$, so that $\gamma = \mu_I$ $(= 2)$.)

Suppose an arbitrary, fixed register R contains 1_G or $\bot$ at a given time. Then, because the chain is normalised,

- The previous operation naming R, if any, initialised it to 1_G;
- The next operation naming R, if any, must be a copy into R;
- If R is for I/O, there is always a previous and a next such operation;
- If R is not for I/O, the first such value has no preceding operation naming R and the final one no such subsequent operation.

Of course, the 1_Gs which occur are in one-to-one correspondence with the chain operations which include an initialisation, each being associated with the operation which created it. Copyings can only overwrite 1_G or $\bot$, and so there is a one-to-one correspondence between copyings (with or without an initialisation) and any instance of $\bot$ or non-final instances of 1_G, each being associated with the copying which destroys it. There is only an instance of $\bot$ if R is not for I/O, and then only one. A final instance of 1_G means one which remains in the register at the end of executing the chain. There can only be one such instance, and it only occurs for a non-IO register. So there is the same number of instances of $\bot$ as number of instances of a final 1_G. Thus the number of times register R is explicitly initialised to 1_G is equal to the number of occurrences of 1_G in R during the execution of the chain, and this in turn equals the number of copyings (with or without initialisation) into R. Summing over all R, $\mu_I+\gamma_I = \gamma+\gamma_I$. So $\mu_I = \gamma$. Since copyings without initialisation become multiplications with initialisation and *vice versa* when the dual is taken, and these numbers are the same, the numbers of them are not changed when the dual is applied. □

5 Preserving the Chain Output under Duality

The action of a space-aware chain $\rho = (\rho_1, \rho_2, \rho_3, \ldots, \rho_n)$ is given by the composition $\nu(\rho) = \rho_n \circ \ldots \circ \rho_3 \circ \rho_2 \circ \rho_1$ of its elements. For a specific chain, this would be calculated as the matrix product, say M_ρ, of the representatives for each operation. The dual chain has action given by the transpose $\nu(\rho^\mathsf{T}) = \rho_1^\mathsf{T} \circ \rho_2^\mathsf{T} \circ \rho_3^\mathsf{T} \circ \ldots \circ \rho_n^\mathsf{T}$.

Definition 7. *A location-aware chain ρ is* symmetric *if $\nu(\rho^\mathsf{T}) = \nu(\rho)$.*

In other words, a symmetric chain is one such that the dual computes the same output. Its matrix is symmetric because the dual is represented by the transpose matrix.

Lemma 1
i) With the above notation for a location-aware chain ρ, $M_\rho^\mathsf{T} = M_{\rho^\mathsf{T}}$.
ii) The chain ρ is symmetric if, and only if, its matrix M_ρ is symmetric.

This gives a criterion for checking whether or not the dual chain will compute the same value(s): it must be symmetric. In the case of a normalised chain, $M_\rho = (m_{ij})$ has $m_{ij} = 0$ if i is the index of a non-I/O register. This is because 1_G is the final value left in register R_i by ρ. Similarly, $m_{ij} = 0$ if j is the index of a non-I/O register. This is because the initial value in register R_i prior to applying ρ is not used by ρ. Hence the action of ρ is entirely described by the sub-matrix of elements indexed by I_{IO}. Consequently,

Theorem 2. *If a normalised, location-aware chain has only one I/O register, then its dual computes the same value.*

Thus, unless we are performing multi-exponentiations, a normalised chain and its dual will certainly output the same values from a given input.

6 Mixed Base Representations

Most exponentiation algorithms start by performing a recoding of the exponent D (normally from binary) into some variety of the *mixed base* form [4,11]:

$$D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \quad \text{with} \quad (r_i, d_i) \in \mathcal{R}\times\mathcal{D} \qquad (1)$$

where $\mathcal{R}$ is a set of allowed *radices*, e.g. $\mathcal{R} = \{2, 4\}$ or $\mathcal{R} = \{2, 3, 5\}$; $\mathcal{D}$ is a set of possible *digits*, such as $\mathcal{D} = \{0, 1, 2, 3, 4\}$, $\mathcal{D} = \{0, 1, 3\}$ or $\mathcal{D} = \{0, \pm1, \pm2\}$; and there are some *rules* on the allowable choices for radix/digit pairs (r_i, d_i), such as a pair of consecutive digits having to include at least one 0 (as in NAF). In general, these representations can be generated by the usual change-of-base algorithm modified to vary the base choice as necessary at each step. As an example, $235_{10} = (((((1)3+0)2+1)5+4)2+0)3+1 = 1_2 0_3 1_2 4_5 0_2 1_3$. A typical step in generating this is to choose base 3 for 235, obtain the (least significant) digit 1_3 as 235 mod 3 and repeat the process on $(235-1)/3 = 78$.

The recoding enables the exponentiation to be simplified into a sequence of easy steps which process the digits from left to right or right to left. Those steps are converted into a space-aware addition chain when implemented. Normalised,

Inputs: $M \in G$, $D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$ where $d_i \in \mathcal{D}$
Output: $M^D \in G$

---

read $P \leftarrow M$
Initialisation: $T[d] \leftarrow P^d$ **for all** $d \neq 0$
$P \leftarrow 1_G$
for $i \leftarrow n-1$ **downto** 0 **do** {
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$
  if $d_i \neq 0$ **then** $P \leftarrow P \times T[d_i]$ }
Finalisation: $T[d] \leftarrow 1_G$ **for all** $d \neq 0$
return P

read $P \leftarrow M$
Initialisation: $T[d] \leftarrow 1_G$ **for all** $d \neq 0$
for $i \leftarrow 0$ **to** $n-1$ **do** {
  if $d_i \neq 0$ **then** $T[d_i] \leftarrow T[d_i] \times P$
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$ }
Finalisation: $P \leftarrow \prod_{d\neq 0} T[d]^d$
$T[d] \leftarrow 1_G$ **for all** $d \neq 0$
return P

Fig. 1. Left-to-Right (left) and Right-to-Left (right) Table-based Exponentiation

but high level, versions of Brauer's m-ary scheme [3] and the scheme of Yao [14] are illustrated in Figure 1. The first line reads the plaintext input M into the only I/O register, namely P, and the last line writes the resulting ciphertext M^D from that register. The non-I/O registers named $T[d], d \in \mathcal{D} \setminus \{0\}$, are initialised before use in the second line, and reduced to a final 1_G in the second last line. The second last line is included to meet the I/O conditions of being normalised; it could be omitted, but is good for security.

Considering just the two registers P and $T[d_i]$ for a fixed i, the matrix corresponding to the addition chain which performs the loop iteration of index i is $\begin{bmatrix} r_i & 1 \\ 0 & 1 \end{bmatrix}$ for the left-to-right version and its transpose, $\begin{bmatrix} r_i & 0 \\ 1 & 1 \end{bmatrix}$ for the right-to-left version. Using Defn. 4, this shows that the composite operations corresponding to the loop bodies are duals of each other if they are written out in corresponding ways using the atomic operations of Defn. 2 — the sequence for one composite operation is transposed to give the sequence for the other.

The combination of the initialisation of table T and setting of P to 1_G in the left-to-right case has a matrix representation indexed by P and the $T[d]$, and it is entirely zero except that the Pth column contains 0 in row P and d in the row for $T[d]$. Its transpose is a matrix with only one non-zero row, namely that of index P and with d in the column for $T[d]$, which is clearly the matrix required to achieve the product which is assigned to P after the loop in the right-to-left case. Thus these two parts of the algorithms are also dual when defined suitably in terms of the atomic operations. Lastly, the finalisation line of the left-to-right case and the initialisation line of the right-to-left case are dual, because both are given by the symmetric matrix, indexed by P and the $T[d]$, which is all zeros except for a 1 as the diagonal entry of index P. This completes a proof that the formulations of the algorithms given in Fig. 1 are, in fact, dual because, in the correspondence, the totality of composite operations in one is reversed and transposed to give the operations of the other.

Strictly speaking, the definition of duality has been extended above to algorithms which are described at the level of composite operations on registers rather than the atomic ones of Defn. 2. However, as in Fig. 1, algorithms are often presented at such a level. This presentation usually has to satisfy the I/O requirements of normalised form in order that the transpose of the initialisation stage of one algorithm yields the finalisation step of the other. This was done for Fig. 1. As composite operations can always be decomposed into segements of a normalised location-aware chain, this extended definition of duality between algorithms means there is an underlying duality in the original sense of Defn. 4.

7 A New Compact Exponentiation Algorithm

In a typical resource-constrained embedded system, there is normally only room for a very small table. This was the motivation for the division chain method of Walter [11], given as the right-to-left algorithm in Figure 2. Typically it uses only three registers: explicit T for the accumulating product and P providing

the right power of the plaintext input for each digit, and implicit working space P' for temporary values. The idea is that pairs (r_i, d_i) in the representation of D have efficient addition chains for r_i which include d_i as an intermediate value so that P^{d_i} can be computed cheaply *en route* to P^{r_i}. As in the table-based algorithms of Fig. 1, the dual left-to-right algorithm given in Fig. 2 is derived simply by reversing and transposing each step. Consequently, a duality proof would follow the same pattern as for the table-based algorithms.

Inputs: $M \in G$, $D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$ where $d_i \in \mathcal{D}$
Output: $M^D \in G$

---

read $P \leftarrow M$
Initialisation: $T \leftarrow P$
$P \leftarrow 1_G$
for $i \leftarrow n-1$ **downto** 0 **do**
if $i \neq n-1$ **then** $P \leftarrow P^{r_i} \times T^{d_i}$
else $P \leftarrow T^{d_i}$
Finalisation: $T \leftarrow 1_G$
return P

read $P \leftarrow M$
Initialisation: $T \leftarrow 1_G$
for $i \leftarrow 0$ **to** $n-1$ **do in parallel** {
$T \leftarrow T \times P^{d_i}$
if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$ }
Finalisation: $P \leftarrow T$
$T \leftarrow 1_G$
return P

Fig. 2. Left-to-Right (left) and Right-to-Left (right) Compact Exponentiation

The dual space-aware chain of atomic operations is still needed for each loop iteration. The right-to-left loop iteration is achieved by a matrix indexed by P and T, namely $\begin{bmatrix} r_i & 0 \\ d_i & 1 \end{bmatrix}$. Its transpose, $\begin{bmatrix} r_i & d_i \\ 0 & 1 \end{bmatrix}$, leads to the dual code given for the left-to-right case. As an example, an iteration with pair $(5, 3)$ can be computed with the addition chain $1{+}1 = 2; 1{+}2 = 3; 2{+}3 = 5$. Using the three registers P, T, and working space P', this can be achieved by the space-aware chain $P \to P'; P' \times P' \to P'; P \times P' \to P; T \times P \to T; P \times P' \to_I P$ where the subscript $_I$ indicates the operation with an initialisation to 1_G. The dual sub-chain is $P \to P'; P \times T \to P; P' \times P \to P'; P' \times P' \to P'; P \times P' \to_I P$. One can readily check that the numbers of each type of operation are the same in this example: one squaring, one multiplication with initialisation, two other multiplications and one copying. Thus there is the effect of having a table which includes M^3 without having to reserve the space for it or spend extra time computing it. Previously it was unclear which digits could be generated this way in a left-to-right algorithm as there was no obvious construction for the required addition sub-chain. Duality solves that problem, as illustrated here with the pair $(5, 3)$.

As the time efficiency is the same for both directions, it is possible to use figures from [12] to see that the algorithm has very similar execution time to the usual algorithms which use similar space (e.g. three registers). When D is fixed for many exponentiations, the cost of the mixed base recoding can be amortised over the lifetime of the key, and the times from [11] apply. This recoding can be biased to make the best use of any composite operations on G, such as a Frobenius map, which are cheaper than their components. Depending upon

where the multiplication by T occurs in the sub-chain, one can also apply one of the double-and-add, triple-and-add or quintuple-and-add formulae for composite elliptic curve operations [6,10,9]. Consequently, the new algorithm appears particularly suitable for SSL servers re-using the same key many times. Of course, the time is the same for both directions only using the coarse measurement of counting doubles and adds on the elliptic curve. Use of the composite operations makes modest but different improvements in time to both directions (*see* [1]).

Finally, a few further words on the efficiency of the code. In order to present symmetric versions of the algorithms, there is some extra copying to have a single register for I/O. This is unnecessary in software, but often required in hardware. So, in Fig. 2, M might have been read directly into T instead of P in the left-to-right algorithm, and, dually, the output returned from T rather than P in the right-to-left algorithm. Deletion of the two copyings would still have left dual algorithms computing the same values, although not symmetric, because the matrix for the computation has a single non-zero value. Lastly, if the rules of normalisation are followed when converting the recoding into a space-aware chain then the multiplications by P^{d_i} or T^{d_i} are automatically removed when $d_i = 0$, rendering unnecessary the condition $d_i \neq 0$ that appeared in Fig. 1.

8 Miscellaneous Space Issues for Dual Chains

Returning to general exponentiation algorithms, there may be cost issues in storing the mixed base representation (1). If D is given in binary but one is allowed to choose a base r_i which is not a power of 2, then the recoding must be done from right to left. This can be done on-the-fly for a right-to-left exponentiation method so that minimal additional storage is required for the recoding. However, the left-to-right algorithm requires the complete recoding to be determined and stored in advance. This may not be able to re-use space occupied by D if the key must be kept, but it makes the left-to-right version use more space. On the other hand, as in Figs. 1 and 2, the initial value of M is normally destroyed in the right-to-left direction, but preserved in the left-to-right direction. So extra storage space for input M may be required to retain it in the right-to-left case.

The way in which registers are used also tends to differ between the two directions. Only P is updated in the example left-to-right algorithms, whereas both P and T are updated in the right-to-left cases. This suggests more data movement is required in right-to-left algorithms, especially if the hardware can only write to memory from one register. Thus, although dual exponentiation schemes nominally use the same time and space, there are often relevant secondary space and data movement issues to consider when duality is used in practice.

9 Conclusion

A straight-forward duality mechanism has been provided for addition chains which enables exponentiation algorithms to process digits of the exponent in either direction. In terms of counts of basic operations on the group in which

exponentiation takes place, and storage for such elements, this mechanism preserves both the time and space usage of an exponentiation scheme. It thereby improves on current methods which only address time issues. Time and space differences between the two directions are mainly confined to the recoding phase of an exponentiation and data preservation. The duality was illustrated using Brauer's and Yao's algorithms, and applied to derive a new, compact left-to-right algorithm. This algorithm can make use of composite elliptic curve operations to achieve very competitive execution speeds, and is useful in both embedded crypto-systems and SSL servers.

References

1. Avanzi, R.M.: Delaying and Merging Operations in Scalar Multiplication: Applications to Curve-Based Cryptosystems. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 203–219. Springer, Heidelberg (2007)
2. Bernstein, D.J.: Pippenger's Exponentiation Algorithm (2002), http://cr.yp.to/papers/pippenger.pdf
3. Brauer, A.: On Addition Chains. Bull. Amer. Math. Soc. 45(10), 736–739 (1939)
4. Dimitrov, V., Cooklev, T.: Two Algorithms for Modular Exponentiation using Non-Standard Arithmetics. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E78-A(1), 82–87 (1995)
5. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Theory and Applications for a Double-Base Number System. In: Proc. ARITH 13, pp. 44–51. IEEE, Monterey (1997)
6. Dimitrov, V.S., Imbert, L., Mishra, P.K.: Efficient and Secure Elliptic Curve Point Multiplication using Double-Base Chains. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
7. Gordon, D.M.: A Survey of Fast Exponentiation Algorithms. Journal of Algorithms 27, 129–146 (1998)
8. Knuth, D.E.: The Art of Computer Programming, 3rd edn. Seminumerical Algorithms, §4.6.3, vol. 2, pp. 465–485. Addison-Wesley (1998)
9. Longa, P., Miri, A.: New Composite Operations and Precomputation Scheme for Elliptic Curve Cryptosystems over Prime Fields. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 229–247. Springer, Heidelberg (2008)
10. Mishra, P.K., Dimitrov, V.: Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication Using Multibase Number Representation. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 390–406. Springer, Heidelberg (2007)
11. Walter, C.D.: Exponentiation using Division Chains. In: Proc. ARITH 13, pp. 92–98. IEEE, Monterey (1997)
12. Walter, C.D.: MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 53–66. Springer, Heidelberg (2002)
13. Walter, C.D.: Sliding Windows Succumbs to Big Mac Attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001)
14. Yao, A.C.-C.: On the Evaluation of Powers. SIAM J. Comput. 5(1), 100–103 (1976)

Optimal Eta Pairing on Supersingular Genus-2 Binary Hyperelliptic Curves

Diego F. Aranha[1,⋆], Jean-Luc Beuchat[2], Jérémie Detrey[3], and Nicolas Estibals[3]

[1] Institute of Computing, University of Campinas
Av. Albert Einstein, 1251, CEP 13084-971, Campinas, Brazil
dfaranha@ic.unicamp.br

[2] Graduate School of Systems and Information Engineering, University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
beuchat@risk.tsukuba.ac.jp

[3] CARAMEL project-team, LORIA, INRIA / CNRS / Nancy Université,
Campus Scientifique, BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France
{jeremie.detrey,nicolas.estibals}@loria.fr

Abstract. This article presents a novel pairing algorithm over supersingular genus-2 binary hyperelliptic curves. Starting from Vercauteren's work on optimal pairings, we describe how to exploit the action of the 2^{3m}-th power Verschiebung in order to reduce the loop length of Miller's algorithm even further than the genus-2 η_T approach.

As a proof of concept, we detail an optimized software implementation and an FPGA accelerator for computing the proposed optimal Eta pairing on a genus-2 hyperelliptic curve over $\mathbb{F}_{2^{367}}$, which satisfies the recommended security level of 128 bits. These designs achieve favourable performance in comparison with the best known implementations of 128-bit-security Type-1 pairings from the literature.

Keywords: Optimal Eta pairing, supersingular genus-2 curve, software implementation, FPGA implementation.

1 Introduction

The Weil and Tate pairings were independently introduced in cryptography by Frey & Rück [18] and Menezes, Okamoto & Vanstone [34] as tools to attack the discrete-logarithm problem on some classes of elliptic curves defined over finite fields. The discovery of constructive properties by Joux [29], Mitsunari, Sakai & Kasahara [37], and Sakai, Oghishi & Kasahara [41] initiated the proposal of an ever-increasing number of protocols based on bilinear pairings: identity-based encryption [10], short signature [12], and efficient broadcast encryption [11], to mention but a few. However, such protocols rely critically on efficient implementations of pairing primitives at high security levels on a wide range of targets.

⋆ This work was performed while the author was visiting University of Waterloo.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 98–115, 2012.

Miller described the first iterative algorithm to compute the Weil and Tate pairings back in 1986 [35,36]. The Tate pairing seems to be more suited to efficient implementations (see for instance [25,30]), and has therefore attracted a lot of interest from the research community. A large number of articles, culminating in the η_T pairing algorithm [5], focused on shortening the loop of Miller's algorithm in the case of supersingular abelian varieties. The Ate pairing, introduced by Hess *et al.* [28] for elliptic curves and by Granger *et al.* [24] in the hyperelliptic case, generalizes the η_T approach to ordinary curves. Eventually, several variants of the Ate pairing aiming at reducing the loop length of Miller's algorithm have been proposed in 2008 [27,31,43].

In this work, we target the AES-128 security level. When dealing with ordinary elliptic curves defined over a prime finite field $\mathbb{F}_p$, the family of curves introduced by Barreto & Naehrig (BN) [6] is a nearly optimal choice for the 128-bit security level. Their embedding degree $k = 12$ perfectly balances the security between the ℓ-torsion and the group of ℓ-th roots of unity, where ℓ is a prime number dividing the cardinality of the curve $\#E(\mathbb{F}_p)$. The latest software implementation results on these curves by Aranha *et al.* report computation times below one millisecond on a single core of an Intel Core i7 processor [1].

Supersingular curves over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ are better suited to hardware implementation, and offer more efficient point doubling and tripling formulae than BN-curves. Moreover, supersingularity allows the use of a distortion map and thus provides Type-1 (or symmetric) pairings [19], which cannot be obtained with ordinary curves. However, the embedding degree of a supersingular elliptic curve is always less than or equal to 6 [34]. As a consequence, the security on the curve is too high with respect to the security of the group of ℓ-th roots of unity, and one has to consider curves defined over very large finite fields. Therefore, most of the hardware accelerators are struggling to achieve the AES-128 level of security (see for instance [9] for a comprehensive bibliography). Software implementations at this security level have for instance been reported in [3,8]. However, the computation of a pairing is at least 6 times faster on a BN curve [7].

To mitigate the effect of the bounded embedded degree, Estibals proposed to consider supersingular elliptic curves over field extensions of moderately-composite degree [17]. Curves are then vulnerable to Weil descent attacks [22], but a careful analysis allowed him to maintain the security above the 128-bit threshold. As a proof of concept, he designed a compact Field-Programmable Gate Array (FPGA) accelerator for computing the Tate pairing on a supersingular elliptic curve defined over $\mathbb{F}_{3^{5\cdot 97}}$. Even though he targeted his architecture to low-resource hardware, his timings are very close to those of software implementations of BN curves.

Yet another way to reduce the size of the base field of the Tate pairing in the supersingular case is to consider a genus-2 binary hyperelliptic curve with embedding degree $k = 12$ [20, 40], which is the solution investigated in this work. We indeed show that, thanks to a novel pairing algorithm, these curves can be actually made very effective in the context of software implementations and hardware accelerators for embedded systems.

This paper is organized as follows: after a general reminder on the hyperelliptic Tate pairing (Section 2) and on the Eta pairing on in the case of those particular curves (Section 3), we describe a novel optimal[1] Eta pairing algorithm that further reduces the loop length of Miller's algorithm compared to the η_T approach [5] (Section 4). We then present an optimized software implementation (Section 5) and a low-area FPGA accelerator (Section 6) for the proposed pairing algorithm. We discuss our results and conclude in Section 7.

2 Background Material and Notations

In this section, we briefly recall a few definitions and results about hyperelliptic curves, and more precisely the Tate pairing on such curves. For more details, we refer the interested reader to [16,24].

2.1 Reminder on Hyperelliptic Curves

Let C be an imaginary nonsingular hyperelliptic curve of genus g defined over the finite field $\mathbb{F}_q$, where $q = p^m$ and p is a prime, and whose affine part is given by the equation $y^2 + h(x)y = f(x)$, where $f,\ h \in \mathbb{F}_q[x]$, $\deg f = 2g + 1$, and $\deg h \leq g$.

For any algebraic extension $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$, we define the set of $\mathbb{F}_{q^d}$-rational points of C as $C(\mathbb{F}_{q^d}) = \{(x, y) \in \mathbb{F}_{q^d} \times \mathbb{F}_{q^d} \mid y^2 + h(x)y = f(x)\} \cup \{P_\infty\}$, where P_∞ is the point at infinity of the curve. For simplicity's sake, we also write $C = C(\overline{\mathbb{F}}_q)$. Additionally, denoting by ϕ_q the q-th power Frobenius morphism $\phi_q : C \to C$, $(x, y) \mapsto (x^q, y^q)$, and $P_\infty \mapsto P_\infty$, note that a point $P \in C$ is $\mathbb{F}_{q^d}$-rational if and only if $\phi_q^d(P) = P$.

We then denote by Jac_C the Jacobian of C, which is an abelian variety of dimension g defined over $\mathbb{F}_q$, and whose elements are represented by the divisor class group of degree-0 divisors $\mathrm{Pic}_C^0 = \mathrm{Div}_C^0 / \mathrm{Princ}_C$. In other words, two degree-0 divisors D and D' belong to the same equivalence class $\overline{D} \in \mathrm{Jac}_C$ if and only if there exists a non-zero rational function $z \in \overline{\mathbb{F}}_q(C)^*$ such that $D' = D + \mathrm{div}(z)$. Naturally extending the Frobenius map to divisors as $\phi_q : \sum_{P \in C} n_P(P) \mapsto \sum_{P \in C} n_P(\phi_q(P))$, we say that D is $\mathbb{F}_{q^d}$-rational if and only if $\phi_q^d(D) = D$.

It can also be shown that any divisor class $\overline{D} \in \mathrm{Jac}_C(\mathbb{F}_{q^d})$ can be uniquely represented by an $\mathbb{F}_{q^d}$-rational reduced divisor $\rho(\overline{D}) = \sum_{i=1}^{r}(P_i) - r(P_\infty)$, with $r \leq g$, $P_i \neq P_\infty$, and $P_i \neq -P_j$ for $i \neq j$, where the negative of a point $P = (x, y)$ is given via the hyperelliptic involution by $-P = (x, -y - h(x))$. In the following, we also denote by $\epsilon(\overline{D}) = \sum_{i=1}^{r}(P_i)$ the effective part of $\rho(\overline{D})$.

Using the Mumford representation, any non-zero $\mathbb{F}_{q^d}$-rational reduced divisor $D = \rho(\overline{D})$ (and therefore any non-zero element of the Jacobian $\mathrm{Jac}_C(\mathbb{F}_{q^d})$) can be associated with a unique pair of polynomials $[u(x), v(x)]$, with $u,\ v \in \mathbb{F}_{q^d}[x]$

[1] Here the "optimal" qualifier is to be understood more as a reference to Vercauteren's work [43] than an actual claim of optimality.

and such that u is monic, $\deg(v) < \deg(u) = r \leq g$, and $u \mid v^2 + vh - f$. Furthermore, given two reduced divisors D_1 and D_2 in Mumford representation, Cantor's algorithm [13] can be used to compute the Mumford representation of $\rho(D_1 + D_2)$, the reduced divisor corresponding to their sum on the Jacobian.

2.2 Hyperelliptic Tate Pairing

Let ℓ be a prime dividing $\#\operatorname{Jac}_C(\mathbb{F}_q)$ and coprime to q. Let also k be the corresponding embedding degree, *i.e.*, the smallest integer such that $\ell \mid q^k - 1$. We denote by $\operatorname{Jac}_C(\mathbb{F}_{q^k})[\ell]$ the $\mathbb{F}_{q^k}$-rational ℓ-torsion subgroup of Jac_C. The Tate pairing on C is then the well-defined, non-degenerate, and bilinear map

$$\langle .,. \rangle_\ell : \operatorname{Jac}_C(\mathbb{F}_{q^k})[\ell] \times \operatorname{Jac}_C(\mathbb{F}_{q^k})/\ell \operatorname{Jac}_C(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell,$$

defined as $\langle \overline{D}_1, \overline{D}_2 \rangle_\ell \equiv f_{\ell,D_1}(D_2)$, where D_1 and D_2 represent the divisor classes $\overline{D}_1$ and $\overline{D}_2$, respectively, with disjoint supports: $\operatorname{supp}(D_1) \cap \operatorname{supp}(D_2) = \emptyset$. Moreover, for any integer n and any $\mathbb{F}_{q^k}$-rational divisor D, the notation $f_{n,D}$ denotes the Miller function in $\mathbb{F}_{q^k}(C)^*$ which is defined (up to a non-zero constant multiple) by its divisor such that $\operatorname{div}(f_{n,D}) = nD - [n]D$, where $[n]D = \rho(nD)$. In the case of the Tate pairing, since $\overline{D}_1 \in \operatorname{Jac}_C[\ell]$, we have $[\ell]D_1 = 0$ and $\operatorname{div}(f_{\ell,D_1}) = \ell D_1$.

So as to obtain a unique value for the Tate pairing, we also define the reduced Tate pairing as $e : (\overline{D}_1, \overline{D}_2) \mapsto \langle \overline{D}_1, \overline{D}_2 \rangle_\ell^{(q^k-1)/\ell} \in \mu_\ell$, with $\mu_\ell \subseteq \mathbb{F}_{q^k}^*$ the subgroup of ℓ-th roots of unity. Note that for any L such that $\ell \mid L \mid q^k - 1$, we also have $e(\overline{D}_1, \overline{D}_2) = \langle \overline{D}_1, \overline{D}_2 \rangle_L^{(q^k-1)/L}$.

Ensuring that there are no elements of order ℓ^2 in $\operatorname{Jac}_C(\mathbb{F}_{q^k})$, we can also show that there is a natural isomorphism between the quotient $\operatorname{Jac}_C(\mathbb{F}_{q^k})/\ell \operatorname{Jac}_C(\mathbb{F}_{q^k})$ and $\operatorname{Jac}_C(\mathbb{F}_{q^k})[\ell]$. We can then identify these two groups, and define the Tate pairing on the domain $\operatorname{Jac}_C(\mathbb{F}_{q^k})[\ell] \times \operatorname{Jac}_C(\mathbb{F}_{q^k})[\ell]$.

The actual computation of the (reduced) Tate pairing is achieved thanks to Miller's algorithm [35,36], which is based on the observation that, for any integer n, n', and for any $\mathbb{F}_{q^k}$-rational divisor D, one can take the function $f_{n+n',D} = f_{n,D} \cdot f_{n',D} \cdot g_{[n]D,[n']D}$, where $g_{[n]D,[n']D} \in \mathbb{F}_{q^k}(C)^*$ is such that $\operatorname{div}(g_{[n]D,[n']D}) = [n]D + [n']D - [n+n']D$. Note that the function $g_{[n]D,[n']D}$ can be explicitly obtained from the computation of $[n+n']D = \rho([n]D + [n']D)$ by Cantor's algorithm. See for instance [24, Algorithm 2] for more details. Therefore, computing $f_{\ell,D_1}(D_2)$ is tantamount to computing $[\ell]D_1$ on $\operatorname{Jac}_C(\mathbb{F}_{q^k})$ by means of any suitable scalar multiplication algorithm (*e.g.*, addition chain or double-and-add) while keeping track of the $g_{[n]D_1,[n']D_1}$ functions given by Cantor's algorithm and evaluating them at the divisor D_2. Miller's algorithm, based on the double-and-add approach, thus has a complexity of $\lfloor \log_2(\ell) \rfloor + \operatorname{wg}(\ell) - 1$ iterations (*i.e.*, evaluations of such $g_{[n]D_1,[n']D_1}$ functions), where $\operatorname{wg}(\ell)$ denotes the Hamming weight of ℓ.

Finally, let u_∞ be an $\mathbb{F}_q$-rational uniformizer at P_∞ (*i.e.*, $\operatorname{ord}_{P_\infty}(u_\infty) = 1$). For any function $z \in \overline{\mathbb{F}}_q(C)^*$, we denote by $\operatorname{lc}_\infty(z) = (u_\infty^{-\operatorname{ord}_{P_\infty}(z)} \cdot z)(P_\infty)$

the leading coefficient of z expressed as a Laurent series in u_∞. Restricting the domain of the Tate pairing to $\overline{D}_1 \in \mathrm{Jac}_C(\mathbb{F}_q)[\ell]$, one can easily check that $\mathrm{lc}_\infty(f_{\ell,D_1}) \in \mathbb{F}_q^*$ with $D_1 = \rho(\overline{D}_1)$. We can then apply [24, Lemma 1] to show that we can simply compute the Tate pairing as $\langle \overline{D}_1, \overline{D}_2 \rangle_\ell = f_{\ell,D_1}(\epsilon(\overline{D}_2))$, as long as $\mathrm{supp}(D_1) \cap \mathrm{supp}(\epsilon(\overline{D}_2)) = \emptyset$. This last condition is ensured by taking $\overline{D}_2 \in \mathrm{Jac}_C(\mathbb{F}_{q^k})[\ell] \setminus \mathrm{Jac}_C(\mathbb{F}_q)[\ell]$.

3 Eta Pairing on Supersingular Genus-2 Binary Curves

3.1 Curve Definition and Basic Properties

In this work, we consider the family of supersingular genus-2 hyperelliptic curves defined over $\mathbb{F}_2$ by the equation $C_d : y^2+y = x^5+x^3+d$, where $d \in \mathbb{F}_2$. Because of their supersingularity, which provides them with a very efficient arithmetic, along with their embedding degree of 12, which is the highest among all supersingular genus-2 curves, these curves are a target of choice for implementing pairing-based cryptography. They have therefore already been studied in this context in several articles [5,14,20,32,39,40].

For m a positive integer coprime to 6, the cardinality L of the Jacobian of C_d over $\mathbb{F}_{2^m}$ is $L = \#\,\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} + \delta 2^{(3m+1)/2} + 2^m + \delta 2^{(m+1)/2} + 1$, where the value of δ is

$$\delta = \begin{cases} (-1)^d & \text{when } m \equiv 1,\ 7,\ 17, \text{ or } 23 \pmod{24}, \text{ and} \\ -(-1)^d & \text{when } m \equiv 5,\ 11,\ 13, \text{ or } 19 \pmod{24}. \end{cases}$$

The embedding degree of C_d is $k = 12$, and $\#\,\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m}) \mid 2^{12m} - 1$. The Tate pairing and its variants will then map into the degree-12 extension $\mathbb{F}_{2^{12m}}$, which we represent as the tower field $\mathbb{F}_{2^{12m}} \cong \mathbb{F}_{2^m}[\tau, s_{\tau,0}]$ where $\tau \in \mathbb{F}_{2^6}$ is such that $\tau^6 + \tau^5 + \tau^3 + \tau^2 + 1 = 0$, and $s_{\tau,0} \in \mathbb{F}_{2^{12}}$ is such that $s_{\tau,0}^2 + s_{\tau,0} + \tau^5 + \tau^3 = 0$.

3.2 Distortion Maps

Since C_d is supersingular, it has non-trivial distortion maps [21,44] embedding $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})$ into distinct subgroups of $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^{12m}})$. Such a distortion map will then allow us to construct Type-1 pairings [19], such as the modified Tate pairing described in the next section. An exhaustive study of the distortion maps of Jac_{C_d} is given by Galbraith *et al.* in [21], of which we now recall the key results.

From [21, Sec. 8], the automorphisms of C_d are of the form

$$\sigma_\omega : (x, y) \mapsto (x + \omega, y + s_{\omega,2}x^2 + s_{\omega,1}x + s_{\omega,0}),$$

where ω is a root of the polynomial $x^{16} + x^8 + x^2 + x$, $s_{\omega,2} = \omega^8 + \omega^4 + \omega$, $s_{\omega,1} = \omega^4 + \omega^2$, and $s_{\omega,0}$ is a root of $y^2 + y + \omega^5 + \omega^3$.

Considering τ as above, we also define $\theta = \tau^4 + \tau^2 + \tau$ and $\xi = \tau^4 + \tau^2$. One easily checks that τ, θ, and ξ are all roots of $x^{16} + x^8 + x^2 + x$. Let us now take $s_{\tau,0}$ as above, along with $s_{\theta,0} = s_{\tau,0} + \tau^5 + \tau^2 + \tau + 1$ and $s_{\xi,0} = \tau^4 + \tau^2$. Verifying

that $s_{\omega,0}^2 + s_{\omega,0} + \omega^5 + \omega^3 = 0$ holds for all $\omega \in \{\tau, \theta, \xi\}$, we can now define the three corresponding automorphisms of C_d, namely σ_τ, σ_θ, and σ_ξ, along with their natural extension to its Jacobian Jac_{C_d}.

From [21, Prop. 8.1], all possible distortion maps can be found in $\mathbb{Z}[\phi_{2^m}, \sigma_\tau, \sigma_\theta]$, where ϕ_{2^m} is the 2^m-th power Frobenius map. Furthermore, $\mathbb{Q}[\phi_{2^m}, \sigma_\tau, \sigma_\theta]$ is a 16-dimensional vector space with the direct sum decomposition

$$\mathbb{Q}[\phi_{2^m}, \sigma_\tau, \sigma_\theta] = \mathbb{Q}(\phi_{2^m}) \oplus \sigma_\tau\mathbb{Q}(\phi_{2^m}) \oplus \sigma_\theta\mathbb{Q}(\phi_{2^m}) \oplus \sigma_\xi\mathbb{Q}(\phi_{2^m}).$$

In other words, the four endomorphisms of Jac_{C_d} 1, σ_τ, σ_θ, and σ_ξ are linearly independent over $\mathbb{Q}(\phi_{2^m})$, and any distortion map can be expressed as a $\mathbb{Q}(\phi_{2^m})$-linear combination of these endomorphisms.

Finally, a tedious computation—which, fortunately, can easily be checked using any computer algebra system—gives the three following equalities over $\mathrm{End}(\mathrm{Jac}_{C_d})$:

$$\begin{aligned}
\phi_{2^m}\sigma_\tau\phi_{2^m}^{-1} &= [2^m]\sigma_\tau\phi_{2^m}^{-2} + [\epsilon 2^{2m}]\sigma_\theta\phi_{2^m}^{-4},\\
\phi_{2^m}\sigma_\theta\phi_{2^m}^{-1} &= [-2^{3m}]\sigma_\theta\phi_{2^m}^{-6}, \text{ and}\\
\phi_{2^m}\sigma_\xi\phi_{2^m}^{-1} &= [2^{4m}]\sigma_\xi\phi_{2^m}^{-8} + [\epsilon 2^{5m}]\phi_{2^m}^{-10},
\end{aligned}$$

where $\epsilon = (-1)^e$ and $e = 0$ when $m \equiv 1$ or $11 \pmod{12}$, and 1 otherwise.

3.3 Modified Tate Pairing on C_d

Let ℓ be a large (odd) prime dividing $L = \#\,\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})$. After ensuring that there are no points of order ℓ^2 in $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^{12m}})$, we can restrict the domain of the Tate pairing to $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] \times \mathrm{Jac}_{C_d}(\mathbb{F}_{2^{12m}})[\ell]$, as detailed in Section 2.2. Using a non-trivial distortion map ψ which maps $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]$ to a subgroup $\psi(\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]) \subset \mathrm{Jac}_{C_d}(\mathbb{F}_{2^{12m}})[\ell]$ such that $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] \cap \psi(\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]) = \{\overline{0}\}$, we can then define the reduced modified Tate pairing as the non-degenerate, bilinear map

$$\begin{aligned}
\hat{e} \;:\; \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] \times \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] &\longrightarrow \mu_\ell \subseteq \mathbb{F}_{2^{12m}}^*\\
(\;\overline{D}_1\;,\;\overline{D}_2\;) &\longmapsto \langle \overline{D}_1, \psi(\overline{D}_2)\rangle_\ell^{(2^{12m}-1)/\ell}\\
&\qquad = \langle \overline{D}_1, \psi(\overline{D}_2)\rangle_L^{(2^{12m}-1)/L},
\end{aligned}$$

where $\langle \overline{D}_1, \psi(\overline{D}_2)\rangle_L = f_{L,D_1}(\epsilon(\psi(D_2)))$, the divisor classes $\overline{D}_1$ and $\overline{D}_2$ being represented by the $\mathbb{F}_{2^m}$-rational reduced divisors $D_1 = \rho(\overline{D}_1)$ and $D_2 = \rho(\overline{D}_2)$. As long as $\overline{D}_1$ and $\overline{D}_2$ are not both trivial, the distortion map ψ ensures that the affine supports of D_1 and $\psi(D_2)$ are disjoint.

At this stage, we have to point out that, in this case, the $g_{[n]D_1,[n']D_1}$ functions required by Miller's algorithm in the computation of the Tate pairing can be simplified. Indeed, from Cantor's algorithm, most of these functions involve vertical lines, which all pass through multiples of the $\mathbb{F}_{2^m}$-rational reduced divisor D_1, meaning that their equations will also be $\mathbb{F}_{2^m}$-rational. Furthermore, noticing that the x-coordinate of $\psi(P)$ is always in $\mathbb{F}_{2^{6m}}$ when P is $\mathbb{F}_{2^m}$- or $\mathbb{F}_{2^{2m}}$-rational, we can conclude that the evaluation of those vertical lines at $\epsilon(\psi(D_2))$

for any $\mathbb{F}_{2^m}$-rational reduced divisor D_2 will also be in $\mathbb{F}^*_{2^{6m}}$ and therefore annihilated by the final exponentiation to the $(2^{12m}-1)/L$-th power. We can then safely ignore the computation of those vertical lines.

3.4 Choosing an Efficient Pairing

Action of the Frobenius ϕ_{2^m}. Following the papers on hyperelliptic Ate and optimal Ate pairings [24, 43], a natural choice is to study the action of ϕ_{2^m}, the 2^m-th power Frobenius map, over $\mathrm{Jac}_{C_d}[\ell]$ in order to reduce the number of iterations in Miller's algorithm.

To that intent, let us first consider a non-zero element $\overline{D}_1 \in \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]$. Since the four endomorphisms 1, σ_τ, σ_θ, and σ_ξ are $\mathbb{Q}(\phi_{2^m})$-linearly independent as per [21, Prop. 8.1], this is also the case for the four elements $\overline{D}_1$, $\overline{D}_\tau = \sigma_\tau(\overline{D}_1)$, $\overline{D}_\theta = \sigma_\theta(\overline{D}_1)$, and $\overline{D}_\xi = \sigma_\xi(\overline{D}_1)$, which then form a basis $\mathcal{B} = (\overline{D}_1, \overline{D}_\tau, \overline{D}_\theta, \overline{D}_\xi)$ of the 4-dimensional ℓ-torsion $\mathrm{Jac}_{C_d}[\ell]$.

From the three equalities presented in Section 3.2, and noting that $\phi_{2^m}(\overline{D}_1) = \overline{D}_1$ since $\overline{D}_1$ is $\mathbb{F}_{2^m}$-rational, one then obtains the following matrix describing the action of ϕ_{2^m} on the ℓ-torsion in the basis $\mathcal{B}$:

$$\phi_{2^m} \equiv \begin{pmatrix} 1 & 0 & 0 & \epsilon 2^{5m} \\ 0 & 2^m & 0 & 0 \\ 0 & \epsilon 2^{2m} & -2^{3m} & 0 \\ 0 & 0 & 0 & 2^{4m} \end{pmatrix} \pmod{\ell}.$$

From this matrix, one can remark that it is not completely diagonal. In particular, the eigenspace of eigenvalue 2^m, which would allow one to construct the optimal Ate pairing described by Vercauteren in [43, Sec. IV-G], is not directly attainable using the distortion map σ_τ. This is not a problem in general, but since we want to construct a Type-1 pairing, we cannot avoid the use of distortion maps.

Diagonalizing the matrix shows that a way to map $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]$ to this eigenspace would be to use the distortion map $\psi = (2^{3m} + \phi_{2^m})\sigma_\tau$, as one can rapidly check that $\phi_{2^m}(\psi(\overline{D}_1)) = [2^m]\psi(\overline{D}_1)$. However, contrary to the distortion maps σ_τ, σ_θ, and σ_ξ which are simple automorphisms of C_d, ψ only acts on its Jacobian. As this might have a negative impact on the performance of the corresponding hyperelliptic Ate pairing, we decide not to follow this option in this paper, even though we plan to investigate it in the near future.

Sticking now to the diagonal parts of the matrix, one might alternatively consider using the distortion map σ_θ, as it maps the $\mathbb{F}_{2^m}$-rational ℓ-torsion to the eigenspace of eigenvalue -2^{3m}. However, since $\ell \mid L \mid 2^{6m}+1$, the lattice in which to look for an optimal pairing over this eigenspace is only of dimension 2, which is no better that the Eta pairing that we propose at the end of this section.

Action of the Verschiebung $\hat{\phi}_{2^m}$. An alternative to relying on the action of the Frobenius map ϕ_{2^m} would be to use its dual $\hat{\phi}_{2^m}$, the 2^m-th power Verschiebung. However, the curve C_d is not superspecial, which means that $\hat{\phi}_{2^m}$,

albeit purely inseparable, is not a map of C_d but only of Jac_{C_d}: the conditions of [24, Lemma 5] are not met, and we are therefore unable to construct a non-degenerate pairing from such a map.

Action of the Verschiebung $\hat{\phi}_{2^{3m}}$. Nevertheless, as already noted by Barreto *et al.* in [5], the 2^{3m}-th power Verschiebung $\hat{\phi}_{2^{3m}}$ can be used instead of $\hat{\phi}_{2^m}$. We detail this construction in the following paragraphs.

First, let $P = (x_P, y_P)$ be a point of C_d distinct from P_∞, and $D = (P) - (P_\infty)$ be the corresponding degenerate divisor. Its Mumford representation is then $D = [x + x_P, y_P]$. Doubling and reducing D three times via Cantor's algorithm, we obtain $[8]D = \rho(8D) = [x + x_P^{64} + 1, x_P^{128} + y_P^{64} + 1]$. Note that the divisor $[8]D$ is also degenerate, as $[8]D = ([8]P) - (P_\infty)$, and corresponds to the point $[8]P = (x_P^{64} + 1, x_P^{128} + y_P^{64} + 1) \in C_d$.

Octupling therefore acts not only on Jac_{C_d} but also on the curve C_d itself, and in fact restricts to a morphism of curves from C_d to itself, defined over $\mathbb{F}_2$ as $[8] = \sigma_1 \phi_8^2$ with σ_1 the automorphism $(x, y) \mapsto (x + 1, x^2 + y + 1)$ and ϕ_8 the 8th power Frobenius map $(x, y) \mapsto (x^8, y^8)$.

Iterating this octupling m times, we obtain the $\mathbb{F}_2$-rational map $[2^{3m}]$ on C_d defined as $[2^{3m}] = \gamma \phi_{2^{3m}}^2$, with $\gamma = \sigma_1^m : (x, y) \mapsto (x + 1, x^2 + y + \nu)$ and $\nu = (m + 1)/2 \bmod 2$. Note that γ, $\phi_{2^{3m}}$, and $[2^{3m}]$ can be naturally extended to Jac_{C_d}, where the latter corresponds to the multiplication by 2^{3m}.

Furthermore, since $\phi_{2^{3m}}$ is a degree-2^{3m} isogeny of Jac_{C_d}, we know that $\hat{\phi}_{2^{3m}} \phi_{2^{3m}} = [2^{3m}]$. Since $[2^{3m}] = \gamma \phi_{2^{3m}}^2$, we then have $\hat{\phi}_{2^{3m}} = \gamma \phi_{2^{3m}}$ and can thus verify that $\hat{\phi}_{2^{3m}}$ is also a degree-2^{3m} purely inseparable endomorphism of the curve C_d. We are therefore in the conditions of [24, Lemma 5], from which we get that, for any reduced divisor D, $\hat{\phi}_{2^{3m}}(D)$ is also reduced and we have the equality of Miller functions (up to a non-zero constant multiple)

$$f_{n,\hat{\phi}_{2^{3m}}(D)} \circ \hat{\phi}_{2^{3m}} = f_{n,D}^{2^{3m}}. \tag{1}$$

Let us now consider the action of $\hat{\phi}_{2^{3m}}$ on the ℓ-torsion $\mathrm{Jac}_{C_d}[\ell]$. Noting that $\phi_{2^{3m}}^4$ is the identity over the ℓ torsion since $\mathrm{Jac}_{C_d}[\ell] \subseteq \mathrm{Jac}_{C_d}(\mathbb{F}_{2^{12m}})$, we obtain the following diagonal matrix in the basis $\mathcal{B}$:

$$\hat{\phi}_{2^{3m}} = [2^{3m}]\phi_{2^{3m}}^{-1} \equiv [2^{3m}]\phi_{2^{3m}}^3 \equiv \begin{pmatrix} 2^{3m} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2^{3m} \end{pmatrix} \pmod{\ell}.$$

From this matrix, it appears that $\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]$ is in the eigenspace of eigenvalue 2^{3m}, while $\psi(\mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell])$ is in the eigenspace of eigenvalue 1, where the distortion map ψ is either σ_τ or σ_θ. In other words, for any $\mathbb{F}_{2^m}$-rational ℓ-torsion element $\overline{D}$, $\hat{\phi}_{2^{3m}}(\overline{D}) = [2^{3m}]\overline{D}$ and $\hat{\phi}_{2^{3m}}(\psi(\overline{D})) = \psi(\overline{D})$.

3.5 Eta Pairing on C_d

We now follow the construction of Barreto *et al.* [5] in order to obtain the η_T pairing with $T = 2^{3m}$. Remarking indeed that $\ell \mid L \mid N$ for $N = 2^{12m} - 1 = T^4 - 1$, and taking $M = N/L$, we can write

$$\hat{e}(\overline{D}_1, \overline{D}_2)^M = f_{L,D_1}(\epsilon(\psi(D_2)))^{M(2^{12m}-1)/L} = f_{N,D_1}(\epsilon(\psi(D_2)))^{(2^{12m}-1)/L}.$$

As $\ell \mid N$, we can then take the Miller function

$$f_{N,D_1} = f_{N+1,D_1} = f_{T^4,D_1} = \prod_{i=0}^{3} f_{T,[T^i]D_1}^{T^{3-i}} = \prod_{i=0}^{3} f_{2^{3m},[2^{i\cdot 3m}]D_1}^{2^{(3-i)\cdot 3m}}.$$

Furthermore, since D_1 and D_2 are $\mathbb{F}_{2^m}$-rational reduced divisors, we also have that $[2^{i\cdot 3m}]D_1 = \hat{\phi}_{2^{3m}}^i(D_1)$ and $\epsilon(\psi(D_2)) = \hat{\phi}_{2^{3m}}^i(\epsilon(\psi(D_2)))$ for all i. Iterating (1) then yields

$$\begin{aligned} f_{2^{3m},[2^{i\cdot 3m}]D_1}(\epsilon(\psi(D_2))) &= \left(f_{2^{3m},\hat{\phi}_{2^{3m}}^i(D_1)} \circ \hat{\phi}_{2^{3m}}^i\right)(\epsilon(\psi(D_2))) \\ &= f_{2^{3m},D_1}(\epsilon(\psi(D_2)))^{2^{i\cdot 3m}}. \end{aligned}$$

Putting it all together, we finally obtain

$$\hat{e}(\overline{D}_1, \overline{D}_2)^M = f_{2^{3m},D_1}(\epsilon(\psi(D_2)))^{4\cdot 2^{3\cdot 3m}\cdot(2^{12m}-1)/L},$$

and, as $\ell \nmid 4 \cdot 2^{3\cdot 3m}$,

$$f_{2^{3m},D_1}(\epsilon(\psi(D_2)))^{(2^{12m}-1)/L} = \hat{e}(\overline{D}_1, \overline{D}_2)^{M\cdot(4\cdot 2^{3\cdot 3m})^{-1} \bmod L}.$$

From the bilinearity and the non-degeneracy of the Tate pairing, we can then conclude that the η_T pairing defined as follows is also bilinear and non-degenerate [5]:

$$\begin{array}{rccccc} \eta_T & : & \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] \times \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] & \longrightarrow & \mu_\ell \subseteq \mathbb{F}_{2^{12m}}^* \\ & & (\quad \overline{D}_1 \quad , \quad \overline{D}_2 \quad) & \longmapsto & f_{2^{3m},D_1}(\epsilon(\psi(D_2)))^{(2^{12m}-1)/L}. \end{array}$$

4 Optimal Eta Pairing on C_d

4.1 Construction and Definition

In order to further decrease the loop length in Miller's algorithm, we adapt in this work the optimal pairing technique as introduced by Vercauteren [43] to the case of the action of the 2^{3m}-th power Verschiebung $\hat{\phi}_{2^{3m}}$ and the Eta pairing detailed in the previous section.

To that intent, let us consider the 2-dimensional lattice spanned by the rows of the matrix

$$\mathfrak{L} = \begin{pmatrix} L & 0 \\ -2^{3m} & 1 \end{pmatrix}.$$

Note that since $\ell \mid L \mid 2^{6m}+1$, we know that $2^{6m} \equiv -1 \pmod{\ell}$, meaning that there is no need to look for 2^{3m}-ary expansions of multiples of L having more than two digits.

A shortest vector of $\mathfrak{L}$ is $[c_0, c_1] = [\delta 2^{(m-1)/2}+1, 2^m + \delta 2^{(m-1)/2}]$, which corresponds to taking the multiple $N' = c_1 2^{3m} + c_0 = M'L$ with $M' = 2^{2m} - \delta 2^{(3m-1)/2} - \delta 2^{(m-1)/2} + 1$.

We then have the M'-th power of the reduced modified Tate pairing

$$\hat{e}(\overline{D}_1, \overline{D}_2)^{M'} = f_{N',D_1}(\epsilon(\psi(D_2)))^{(2^{12m}-1)/L},$$

for which we can take the Miller function

$$\begin{aligned} f_{N',D_1} &= f_{c_1 2^{3m},D_1} \cdot f_{c_0,D_1} \cdot g_{[c_0]D_1,[c_1 2^{3m}]D_1} \\ &= f_{2^{3m},D_1}^{c_1} \cdot f_{c_1,[2^{3m}]D_1} \cdot f_{c_0,D_1} \cdot g_{[c_0]D_1,[c_1 2^{3m}]D_1}. \end{aligned}$$

Remarking that $c_1 2^{3m} \equiv -c_0 \pmod{\ell}$, $g_{[c_0]D_1,[c_1 2^{3m}]D_1}$ actually corresponds to the vertical lines passing through $[c_0]D_1$ and $[-c_0]D_1$, which can simply be ignored. Furthermore, exploiting the action of the Verschiebung $\hat{\phi}_{2^{3m}}$, we can rewrite $f_{c_1,[2^{3m}]D_1}(\epsilon(\psi(D_2)))$ as $f_{c_1,D_1}^{2^{3m}}(\epsilon(\psi(D_2)))$. Finally, also note that $f_{2^{3m},D_1}(\epsilon(\psi(D_2)))^{c_1 \cdot (2^{12m}-1)/L}$ is actually a power of the Eta pairing $\eta_T(\overline{D}_1, \overline{D}_2)$ defined in the previous section.

Consequently, let $\eta_{[c_0,c_1]} : \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] \times \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell] \to \mu_\ell$ be the optimal Eta pairing defined as

$$\eta_{[c_0,c_1]} \quad : \quad (\overline{D}_1, \overline{D}_2) \quad \longmapsto \quad \left(f_{c_1,D_1}^{2^{3m}} \cdot f_{c_0,D_1}\right)(\epsilon(\psi(D_2)))^{(2^{12m}-1)/L}.$$

From the previous considerations, we thus have that

$$\hat{e}(\overline{D}_1, \overline{D}_2)^{M'} = \eta_{[c_0,c_1]}(\overline{D}_1, \overline{D}_2) \cdot \eta_T(\overline{D}_1, \overline{D}_2)^{c_1},$$

whence $\eta_{[c_0,c_1]}(\overline{D}_1, \overline{D}_2) = \hat{e}(\overline{D}_1, \overline{D}_2)^W$ with

$$\begin{aligned} W &= M' - c_1 M \cdot (4 \cdot 2^{3 \cdot 3m})^{-1} \bmod L \\ &= 2^{2m} + \delta 2^{(3m-1)/2} + 2^m + \delta 2^{(m-1)/2} + 1. \end{aligned}$$

Finally, as $\ell \nmid W$, we show that the optimal Eta pairing $\eta_{[c_0,c_1]}$ is also bilinear and non-degenerate.

Note that the η_T pairing introduced in [5] with $T = -\delta 2^{(3m+1)/2} - 1$ corresponds to the lattice vector $[-\delta 2^{(3m+1)/2} - 1, -1] \in \mathfrak{L}$.

4.2 Computing $\eta_{[c_0,c_1]}$

The computation of the optimal Eta pairing $\eta_{[c_0,c_1]}$ defined in the previous section relies on the evaluation of the two Miller functions f_{c_0,D_1} and f_{c_1,D_1} at $\epsilon(\psi(D_2))$. With $[c_0, c_1] = [\delta 2^{(m-1)/2}+1, 2^m + \delta 2^{(m-1)/2}]$, we can take the following functions

$$\begin{cases} f_{c_0,D_1} = f_{\delta 2^{(m-1)/2},D_1} \cdot g_{[\delta 2^{(m-1)/2}]D_1,D_1} & \text{and} \\ f_{c_1,D_1} = f_{2^m,D_1} \cdot f_{\delta 2^{(m-1)/2},D_1} \cdot g_{[2^m]D_1,[\delta 2^{(m-1)/2}]D_1}. \end{cases}$$

Since we are ignoring the vertical lines, we can further rewrite

$$f_{\delta 2^{(m-1)/2},D_1} = f_{2^{(m-1)/2},[\delta]D_1} \quad \text{and}$$
$$f_{2^m,D_1} = f_{\delta 2^{(m-1)/2}.\delta 2^{(m+1)/2},D_1} = f^{\delta 2^{(m+1)/2}}_{2^{(m-1)/2},[\delta]D_1} \cdot f_{2^{(m+1)/2},[2^{(m-1)/2}]D_1},$$

which finally gives

$$\begin{cases} f_{c_0,D_1} = f_{2^{(m-1)/2},[\delta]D_1} \cdot g_{[\delta 2^{(m-1)/2}]D_1,D_1} & \text{and} \\ f_{c_1,D_1} = f^{\delta 2^{(m+1)/2}+1}_{2^{(m-1)/2},[\delta]D_1} \cdot f_{2^{(m+1)/2},[2^{(m-1)/2}]D_1} \cdot g_{[2^m]D_1,[\delta 2^{(m-1)/2}]D_1}. \end{cases}$$

The computation of $\eta_{[c_0,c_1]}$ therefore chiefly involves the evaluation of the two Miller functions $f_{2^{(m-1)/2},[\delta]D_1}$ and $f_{2^{(m+1)/2},[2^{(m-1)/2}]D_1}$ of loop length $(m-1)/2$ and $(m+1)/2$, respectively. This represents a saving of 33% with respect to the η_T pairing presented in [5] whose Miller's loop length is $(3m+1)/2$.

Note that in order to exploit the octupling formula, we have to consider two cases, depending on the value of m mod 6, as described in Algorithm 1.

- When $m \equiv 1 \pmod 6$, then $(m-1)/2$ is a multiple of 3, and $f_{2^{(m-1)/2},[\delta]D_1}$ can be computed via $(m-1)/6$ octuplings, whereas $f_{2^{(m+1)/2},[2^{(m-1)/2}]D_1}$ can be computed by means of another $(m-1)/6$ octuplings and one extra doubling.
- When $m \equiv 5 \pmod 6$, $(m-1)/2$ is not a multiple of 3, but $(m+1)/2$ is. We then compute $\eta^2_{[c_0,c_1]} = \eta_{[2c_0,2c_1]}$ instead, with the Miller functions

$$\begin{cases} f_{2c_0,D_1} = f_{2^{(m+1)/2},[\delta]D_1} \cdot f_{2,D_1} \cdot g_{[\delta 2^{(m+1)/2}]D_1,[2]D_1} & \text{and} \\ f_{2c_1,D_1} = f^{\delta 2^{(m+1)/2}+1}_{2^{(m+1)/2},[\delta]D_1} \cdot f_{2^{(m+1)/2},[2^{(m+1)/2}]D_1} \cdot g_{[2^{m+1}]D_1,[\delta 2^{(m+1)/2}]D_1}. \end{cases}$$

  The two $f_{2^{(m+1)/2},D}$ functions are then evaluated using $(m+1)/6$ octuplings each, whereas f_{2,D_1} only require one doubling.

Finally, one should note that, in our case, since the curve C_d is supersingular, the final exponentiation step is much simpler than for ordinary curves such as BN curves. Indeed, the exponent is

$$(2^{12m}-1)/L = (2^{6m}-1)(2^{2m}+1)(2^{2m} - \delta 2^{(3m+1)/2} + 2^m - \delta 2^{(m+1)/2} + 1),$$

whose regular form can be exploited to devise an efficient *ad-hoc* exponentiation algorithm, of negligible complexity when compared to Miller's loop.

4.3 Evaluation of the Complexity

From the above description of the optimal Eta pairing $\eta_{[c_0,c_1]}$, we can see that most of its computational cost lies in the iterated octuplings of D_1 and the evaluation of the corresponding Miller functions of the form $f_{8,[\pm 8^i]D_1}$ at the effective divisor $\epsilon(\psi(D_2))$. Here, we denote by $[\pm 8^i]D_1$ a reduced divisor representing one of the iterated octuples of D_1 or of $[\delta]D_1$ as required in the evaluation of $\eta_{[c_0,c_1]}$.

Algorithm 1. Computation of the optimal Eta pairing

Input: $\overline{D}_1$ and $\overline{D}_2 \in \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m})[\ell]$ represented by the reduced divisors D_1 and D_2.
Output: $\eta_{[c_0,c_1]}(\overline{D}_1, \overline{D}_2)$ or $\eta_{[c_0,c_1]}(\overline{D}_1, \overline{D}_2)^2 \in \mu_\ell \subseteq \mathbb{F}^*_{2^{12m}}$, depending upon whether $m \equiv 1$ or $5 \pmod 6$, respectively.

1. **if** $m \equiv 1 \pmod 6$ **then** $m' \leftarrow m-1$ **else** $m' \leftarrow m+1$ **end if**
2. $G_1 \leftarrow 1$; $R_1 \leftarrow [\delta]D_1$; $E_2 \leftarrow \epsilon(\psi(D_2))$
3. **for** $i \leftarrow 1$ **to** $m'/6$ **do**
4. $\quad G_1 \leftarrow G_1^8 \cdot f_{8,R_1}(E_2)$
5. $\quad R_1 \leftarrow [8]R_1$
6. **end for** // $G_1 = f_{2^{m'/2},[\delta]D_1}(E_2)$ and $R_1 = [\delta 2^{m'/2}]D_1$.
7. $G_2 \leftarrow G_1^\delta$; $R_2 \leftarrow [\delta]R_1$
8. **for** $i \leftarrow 1$ **to** $m'/6$ **do**
9. $\quad G_2 \leftarrow G_2^8 \cdot f_{8,R_2}(E_2)$
10. $\quad R_2 \leftarrow [8]R_2$
11. **end for** // $G_2 = f_{2^{m'},D_1}(E_2)$ and $R_2 = [2^{m'}]D_1$.
12. **if** $m \equiv 1 \pmod 6$ **then**
13. $\quad G_2 \leftarrow G_2^2 \cdot f_{2,R_2}(E_2)$ // $G_2 = f_{2^m,D_1}(E_2)$.
14. $\quad F_0 \leftarrow G_1 \cdot g_{R_1,D_1}(E_2)$ // $F_0 = f_{c_0,D_1}(E_2)$.
15. $\quad F_1 \leftarrow G_1 \cdot G_2 \cdot g_{[2]R_2,R_1}(E_2)$ // $F_1 = f_{c_1,D_1}(E_2)$.
16. **else**
17. $\quad F_0 \leftarrow G_1 \cdot f_{2,D_1}(E_2) \cdot g_{R_1,[2]D_1}(E_2)$ // $F_0 = f_{2c_0,D_1}(E_2)$.
18. $\quad F_1 \leftarrow G_1 \cdot G_2 \cdot g_{R_2,R_1}(E_2)$ // $F_1 = f_{2c_1,D_1}(E_2)$.
19. **end if**
20. **return** $\left(F_1^{2^{3m}} \cdot F_0\right)^{(2^{12m}-1)/L}$

In that sense, since D_1 is defined over $\mathbb{F}_{2^m}$, then $[\pm 8^i]D_1$ is also $\mathbb{F}_{2^m}$-rational. Moreover, as octupling directly acts on the curve C_d, if D_1 is degenerate (*i.e.*, of the form $D_1 = (P) - (P_\infty)$), then so is $[\pm 8^i]D_1$. Finally, note that if D_2 is degenerate, then so is $\psi(D_2)$, meaning that $\epsilon(\psi(D_2))$ is of degree 1 and has only one point in its support.

Considering the Miller function for octupling, we rewrite $f_{8,D} = f_{4,D}^2 \cdot f_{2,[4]D}$. Each iteration of Miller's algorithm is then just a matter of evaluating $f_{4,[\pm 8^i]D_1}$ and $f_{2,[\pm 4 \cdot 8^i]D_1}$ at $\epsilon(\psi(D_2))$, squaring the former, and accumulating both into the running product via two successive multiplications[2] over $\mathbb{F}_{2^{12m}}$. The respective costs of these operations are given in terms of basic operations over the base field $\mathbb{F}_{2^m}$ in Table 1.

Note that in order to obtain these costs, we have constructed $\mathbb{F}_{2^{12m}}$ as the tower field $\mathbb{F}_{2^m}[i, \tau, s_{\tau,0}]$, where $i \in \mathbb{F}_{2^2}$ is such that $i^2 + i + 1 = 0$, $\tau \in \mathbb{F}_{2^6}$ is such that $\tau^3 + i\tau^2 + i\tau + i = 0$ (one can then check that we still have $\tau^6 + \tau^5 + \tau^3 + \tau^2 + 1 = 0$), and $s_{\tau,0}$ is defined as before. Using Karatsuba for the two quadratic extensions and Toom–Cook for the cubic one, we obtain the expected complexity of 45 multiplications over $\mathbb{F}_{2^m}$ for computing one product over $\mathbb{F}_{2^{12m}}$ [30].

[2] Note that these multiplications are sparser than a regular multiplication over $\mathbb{F}_{2^{12m}}$ if at least one of the two divisors D_1 or D_2 is degenerate.

Where relevant, several costs are given in Table 1, depending on whether D_1 and D_2 are general (Gen.) or degenerate (Deg.) divisors. Making this disctinction is particularly relevant, as some protocols might be able to constrain the domain of their pairing computations in order to benefit from a possible speedup of 2 when one argument is degenerate, or even 4 in the case of two. For instance, Chatterjee *et al.* [14] have proposed a variant of the BLS signature scheme [12] in which one argument of each pairing function is a degenerate divisor.

Table 1. Costs of various operations involved in the computation of the optimal Eta pairing in terms of basic operations (multiplication, addition, squaring, and inversion) over the base field $\mathbb{F}_{2^m}$

Operation	D_1	D_2	Operations over $\mathbb{F}_{2^m}$ Mult.	Add.	Sq.	Inv.
Addition over $\mathbb{F}_{2^{12m}}$	—	—	0	12	0	0
Squaring over $\mathbb{F}_{2^{12m}}$	—	—	0	21	12	0
Multiplication over $\mathbb{F}_{2^{12m}}$	—	—	45	199	0	0
$[\pm 8^i]D_1 \mapsto [\pm 8^{i+1}]D_1$	Deg.	—	0	2	13	0
	Gen.	—	0	5	24	0
$f_{4,[\pm 8^i]D_1}(\epsilon(\psi(D_2)))$	Deg.	Deg.	3	11	1	0
	Gen.	Deg.	19	40	2	0
	Gen.	Gen.	83	247	17	0
$f_{2,[\pm 4\cdot 8^i]D_1}(\epsilon(\psi(D_2)))$	Deg.	Deg.	2	9	1	0
	Gen.	Deg.	16	34	2	0
	Gen.	Gen.	81	236	17	0
Miller iteration $\begin{cases} G_i \leftarrow G_i^8 \cdot f_{8,R_i}(E_2) \\ R_i \leftarrow [8]R_i \end{cases}$	Deg.	Deg.	61	315	68	0
	Gen.	Deg.	121	512	130	0
	Gen.	Gen.	254	949	160	0
Final exp. over $\mathbb{F}_{2^{367}}$	—	—	303	1 386	2 234	1
Optimal Eta pairing $\eta_{[c_0,c_1]}(\overline{D}_1, \overline{D}_2)$ over $C_0(\mathbb{F}_{2^{367}})$	Deg.	Deg.	7 894	40 356	11 571	1
	Gen.	Deg.	15 293	64 644	15 472	1
	Gen.	Gen.	31 644	118 382	19 161	1

In the two following sections, as a proof of concept, we detail the software and hardware implementation results of the proposed optimal Eta pairing $\eta_{[c_0,c_1]}$. The selected curve is C_0 (*i.e.*, $d = 0$) over the field $\mathbb{F}_{2^{367}}$. One can check that $\#\,\mathrm{Jac}_{C_0}(\mathbb{F}_{2^{367}}) = 13 \cdot 7170258097 \cdot \ell$, where ℓ is a 698-bit prime, while the finite field $\mathbb{F}_{2^{12\cdot 367}}$ ensures a security of 128 bits for the computation of discrete logarithms via the function field sieve. The costs of the optimal Eta pairing on $C_0(\mathbb{F}_{2^{367}})$ are also given in Table 1.

For comparison purposes, one might compare this with the costs for the η_T pairing over C_d presented in [14] and [32]. In the former, Chatterjee *et al.* report a cost of 15 111 $\mathbb{F}_{2^m}$-multiplications for an η_T pairing on two degenerate divisors over $C_0(\mathbb{F}_{2^{459}})$. Since the number of these multiplications scales linearly with the size of the field, their approach would entail roughly 12 000 multiplications over our curve. In [32], Lee and Lee require 11 488 multiplications for an η_T pairing on two general divisors over $C_d(\mathbb{F}_{2^{79}})$, which would scale to approximately 53 000 multiplications on our curve $C_0(\mathbb{F}_{2^{367}})$. When compared to the figures in Table 1, these costs reflect the 33% improvement achieved thanks to our proposed optimal Eta approach.

5 Software Implementation

A software implementation was realized to illustrate the performance of the proposed pairing. The C programming language was used in conjunction with compiler intrinsics for accessing vector instructions. The chosen compiler was GCC version 4.6.2 with compiler flags including optimization level `-O3`, loop unrolling and platform-dependent tuning with `-march=native`. For evaluation, we considered as target platforms the Core 2 Duo 45 nm (Penryn microarchitecture) and Core i5 32 nm (Nehalem microarchitecture), represented by an Intel Xeon X3320 2.5 GHz and a mobile Intel Core i5 540 2.53 GHz with Turbo Boost disabled, respectively. Field arithmetic was implemented following the vectorization-friendly formulation presented in [2], with the exception of the Core i5 platform, where multiplication in $\mathbb{F}_{2^{367}}$ was implemented with the help of the native binary field multiplier [26] following the guidelines suggested in [42], that is, a 128-bit granular organization consisting of 3-way and 2-way Karatsuba formulas. We obtained timings of 7, 41, 464 and 11162 cycles for addition, squaring, multiplication and inversion in the Core 2, respectively; and efficiency gains of 47% and 27% for multiplication and inversion in the Core i5, respectively.

Table 2 presents our timings in millions of cycles for the pairing computation at the 128-bit security level. Timings from several related works are also collected for direct comparison with our software implementation. Our implementation considers all the three possible choices of divisors: general × general (GG), general × degenerate (GD) and degenerate × degenerate (DD); and presents the proposed genus-2 optimal Eta pairing as a very efficient candidate among the Type-1 pairings defined on supersingular curves over small-characteristic fields. In particular, the proposed pairing is more efficient than all other Type-1 pairings when at least one of the arguments is degenerate. Considering the Nehalem microarchitecture as a trend for future 64-bit computing platforms, the proposed pairing computed with degenerate divisors is also the closest in terms of performance to the current speed record for Type-3 pairing computation [1].

Table 2. Software implementations of pairing at the 128-bit security level. Timings were obtained with the Turbo Boost feature turned off, and therefore are compatible with the timings in Table 4 of the extended version of [1].

Implementation	Curve	Pairing	Intel Core 2 ($\times 10^6$ cycles)	Intel Core Nehalem ($\times 10^6$ cycles)
Beuchat *et al.* [8]	$E(\mathbb{F}_{2^{1223}})$	η_T	23.03	—
	$E(\mathbb{F}_{3^{359}})$		15.13	—
Aranha *et al.* [3], [4]	$E(\mathbb{F}_{2^{1223}})$	η_T	18.76	8.28
Chatterjee *et al.* [14]	$E(\mathbb{F}_{2^{1223}})$	η_T	19.0	—
	$E(\mathbb{F}_{3^{359}})$		15.8	—
	$C_0(\mathbb{F}_{2^{439}})$	η_T (DD)	16.4	—
Naehrig *et al.* [38]	$E(\mathbb{F}_p)$	Opt. Ate	4.38	—
Beuchat *et al.* [7]	$E(\mathbb{F}_p)$	Opt. Ate	2.95	2.82*
Aranha *et al.* [1]	$E(\mathbb{F}_p)$	Opt. Ate	2.19	2.04*
This work	$C_0(\mathbb{F}_{2^{367}})$	**Opt. Eta (DD)**	**4.44**	**2.75**
		Opt. Eta (GD)	**8.37**	**5.04**
		Opt. Eta (GG)	**16.95**	**9.90**

*Results adjusted by the maximum overclocking rate to eliminate the effect of Turbo Boost.

6 FPGA Implementation

We detail here an FPGA accelerator for our optimal Eta pairing on the curve $C_0(\mathbb{F}_{2^{367}})$ when both inputs are general divisors (GG). In [9], Beuchat *et al.* have presented a coprocessor architecture for computing the final exponentiation of the η_T pairing over supersingular curves. The core of their arithmetic and logic unit is a parallel–serial multiplier processing D coefficients of the multiplicand at each clock cycle, along with a unified operator supporting addition, Frobenius map, and n-fold Frobenius map. Intermediate results are stored in a register file implemented by means of dual-ported RAM. We decided to adapt such a finite field coprocessor for implementing our optimal Eta pairing. In the case of the finite field $\mathbb{F}_{2^{367}}$, we selected the parameters $D = 16$ and $n = 3$ for this coprocessor. We prototyped our architecture on several Xilinx FPGAs with average speedgrade (Table 3). Place-and-route results show for instance that our pairing accelerator uses 4518 slices and 20 RAM blocks of a Virtex-4 device clocked at 220 MHz. For comparison purposes, we also included recent hardware implementation results from the literature in Table 3. It appears that our design is very compact and that its computation time remains comparable to other 128-bit-security implementations. This is even more so when noting that our performance estimates are given for the pairing of two general divisors, and that a speedup of 2 or 4 might be expected from the use of one or two degenerate divisors, respectively.

Table 3. FPGA implementations of pairings at medium- and high-security levels

Implementation	Curve	Sec. (bits)	FPGA	Area (slices)	Freq. (MHz)	Time (μs)	Area×time (slices·s)
Ronan *et al.* [39]	$C_0(\mathbb{F}_{2^{103}})$ (DD)	75	xc2vp100-6	30464	41	132	4.02
Beuchat *et al.* [9]	$E(\mathbb{F}_{2^{691}})$	105	xc4vlx200-11	78874	130	19	1.48
	$E(\mathbb{F}_{3^{313}})$	109	xc4vlx200-11	97105	159	17	1.64
Cheung *et al.* [15]	$E(\mathbb{F}_{p_{254}})$	126	xc6vlx240t-2	7032*	250	573	4.03
Ghosh *et al.* [23]	$E(\mathbb{F}_{2^{1223}})$	128	xc4vlx200-11	35458	168	286	10.14
			xc6vlx130t-3	15167	250	190	2.88
Estibals [17]	$E(\mathbb{F}_{3^{5\cdot 97}})$	128	xc4vlx25-11	4755	192	2227	10.59
			xc3s1000-5	4713	104	4113	19.38
This work	$\boldsymbol{C_0(\mathbb{F}_{2^{367}})}$ **(GG)**	**128**	**xc2vp30-6**	**4646**	**176**	**4405**	**20.5**
			xc4vlx25-11	**4518**	**220**	**3518**	**15.9**
			xc3s1500-5	**4713**	**114**	**6800**	**32.0**

*Number of Virtex-6 slices; this design also uses 32 embedded DSP blocks.

7 Conclusion and Perspectives

We presented a novel optimal Eta pairing algorithm on supersingular genus-2 binary hyperelliptic curves. Starting from Vercauteren's work on optimal pairings [43], we described how to exploit the action of the 2^{3m}-th power Verschiebung in order to further reduce the loop length of Miller's algorithm with respect to the genus-2 η_T approach [5], thus resulting in a 33% improvement.

In order to demonstrate the efficiency of our approach, we implemented the optimal Eta pairing at the 128-bit security level in software and hardware. As far as Type-1 pairings are concerned, our results show that genus-2 curves are a very effective alternative to supersingular elliptic curves and can even compete with the Type-3 pairings provided by ordinary curves such as BN curves.

We have designed as well an FPGA coprocessor for computing the proposed pairing, which also compares very well against other hardware pairing implementations. Additionally, this is the first known hardware pairing implementation over a genus-2 hyperelliptic curve reaching 128 bits of security.

Building upon this work, we now plan to study more precisely the action of other purely inseparable maps on C_d along with the corresponding pairing algorithms, so as to identify which one is the most efficient from an implementation point of view. Indeed, apart from the presented optimal Eta pairing based on the action of $\hat{\phi}_{2^{3m}}$, one can also construct optimal Ate pairings using the action of $\phi_{2^{3m}}$, or that of ϕ_{2^m} under the distortion map σ_θ, the most promising candidate being the optimal Ate pairing for the action of ϕ_{2^m} under the distortion map $\psi = (2^{3m} + \phi_{2^m})\sigma_\tau$.

Furthermore, Lubicz & Robert have recently presented a novel technique for computing the Weil and Tate pairings over abelian varieties based on an efficient representation of their elements by means of theta functions [33]. We are planning to investigate the application of this method to the case of our proposed genus-2 optimal Eta pairing, as both software and hardware implementations might benefit from the faster arithmetic of theta functions.

Acknowledgments. First of all, the authors would like to express their deepest thanks to Guillaume Hanrot who advised us to have a go at genus-2 pairings. He shall receive here our utmost gratitude. The authors would also like to thank Pierrick Gaudry for the careful proof-reading of the technical sections of this paper, along with Gaëtan Bisson, Romain Cosset, and Emmanuel Thomé who were always available to provide some clear answers to our many questions. Last but definitely not least, the authors would like to thank the anonymous reviewers for their insightful comments and suggestions for improving this paper.

References

1. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer (2011)
2. Aranha, D.F., López, J., Hankerson, D.: Efficient software implementation of binary field arithmetic using vector instruction sets. In: Abdalla, M., Barreto, P. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 144–161. Springer (2010)
3. Aranha, D.F., López, J., Hankerson, D.: High-speed parallel software implementation of the η_T pairing. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 89–105. Springer (2010)
4. Aranha, D.F., Menezes, A., Knapp, E., Rodríguez-Henríquez, F.: Parallelizing the Weil and Tate pairings. In: IMA-CC (2011), to appear.

5. Barreto, P., Galbraith, S., Ó Éigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular Abelian varieties. Des. Codes Crypt. 42, 239–271 (2007)
6. Barreto, P., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer (2006)
7. Beuchat, J.L., Díaz, J.G., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over Barreto–Naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. pp. 21–39. No. 6487 in LNCS, Springer (2010)
8. Beuchat, J.L., López-Trejo, E., Martínez-Ramos, L., Mitsunari, S., Rodríguez-Henríquez, F.: Multi-core implementation of the Tate pairing over supersingular elliptic curves. In: Garay, J., Miyaji, A., Otsuka, A. (eds.) CANS 2009. pp. 413–432. No. 5888 in LNCS, Springer (2009)
9. Beuchat, J.L., Detrey, J., Estibals, N., Okamoto, E., Rodríguez-Henríquez, F.: Fast architectures for the η_T pairing over small-characteristic supersingular elliptic curves. Bruguera, J., Cornea, M., Das Sarma, D. (eds.) IEEE Trans. Comput. 60(2), 266–281 (2011)
10. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. pp. 213–229. No. 2139 in LNCS, Springer (2001)
11. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. pp. 258–275. No. 3621 in LNCS, Springer (2005)
12. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. pp. 514–532. No. 2248 in LNCS, Springer (2001)
13. Cantor, D.: Computing in the Jacobian of a hyperelliptic curve. Math. Comput. 48(177), 95–101 (1987)
14. Chatterjee, S., Hankerson, D., Menezes, A.: On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings. In: Hasan, M., Helleseth, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 114–134. Springer (2010)
15. Cheung, R., Duquesne, S., Fan, J., Guillermin, N., Verbauwhede, I., Yao, G.: FPGA implementation of pairings using residue number system and lazy reduction. In: Preneel, B., Takagi, T. (eds.) CHES 2011. pp. 421–441. No. 6917 in LNCS, Springer (2011)
16. Cohen, H., Frey, G. (eds.): Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications, Chapman & Hall/CRC (2006)
17. Estibals, N.: Compact hardware for computing the Tate pairing over 128-bit-security supersingular curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. pp. 397–416. No. 6487 in LNCS, Springer (2010)
18. Frey, G., Rück, H.G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comput. 62(206), 865–874 (1994)
19. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Discrete Applied Mathematics 156, 3113–3121 (2008)
20. Galbraith, S.: Supersingular curves in cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. pp. 495–513. No. 2248 in LNCS, Springer (2001)
21. Galbraith, S.D., Pujolàs, J., Ritzenthaler, C., Smith, B.: Distortion maps for genus two curves. J. Math. Cryptol. 3(1), 1–18 (2009)
22. Gaudry, P., Hess, F., Smart, N.: Constructive and destructive facets of Weil descent on elliptic curves. J. Cryptol. 15(1), 19–46 (2001)
23. Ghosh, S., Roychowdhury, D., Das, A.: High speed cryptoprocessor for η_T pairing on 128-bit secure supersingular elliptic curves over characteristic two fields. In: Preneel, B., Takagi, T. (eds.) CHES 2011. pp. 442–458. No. 6917 in LNCS, Springer (2011)

24. Granger, R., Hess, F., Oyono, R., Thériault, N., Vercauteren, F.: Ate pairing on hyperelliptic curves. In: Naor, M. (ed.) EUROCRYPT 2007. pp. 430–447. No. 4515 in LNCS, Springer (2007)
25. Granger, R., Page, D., Smart, N.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS VII. pp. 480–494. No. 4076 in LNCS, Springer (2006)
26. Gueron, S., Kounavis, M.E.: Carry-less multiplication and its usage for computing the GCM mode. White paper (2010), http://software.intel.com/file/24918
27. Hess, F.: Pairing lattices. In: Galbraith, S., Paterson, K. (eds.) Pairing 2008. pp. 18–38. No. 5209 in LNCS, Springer (2008)
28. Hess, F., Smart, N., Vercauteren, F.: The Eta pairing revisited. IEEE Trans. Inf. Theory 52(10), 4595–4602 (2006)
29. Joux, A.: A one round protocol for tripartite Diffie–Hellman. In: Bosma, W. (ed.) ANTS IV. pp. 385–394. No. 1838 in LNCS, Springer (2000)
30. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N. (ed.) IMA-CC. pp. 13–36. No. 3796 in LNCS, Springer (2005)
31. Lee, E., Lee, H.S., Park, C.M.: Efficient and generalized pairing computation on abelian varieties (2009)
32. Lee, E., Lee, Y.: Tate pairing computation on the divisors of hyperelliptic curves of genus 2. J. Korean Math. Soc. 45(4), 1057–1073 (2008)
33. Lubicz, D., Robert, D.: Efficient pairing computation with Theta functions. In: Hanrot, G., Morain, F., Thomé, E. (eds.) ANTS IX. LNCS, vol. 6197, pp. 251–269. Springer (2010)
34. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curves logarithms to logarithms in a finite field. IEEE Trans. Inf. Theory 39(5), 1639–1646 (1993)
35. Miller, V.: Short programs for functions on curves (1986), unpublished manuscript available at http://crypto.stanford.edu/miller
36. Miller, V.: The Weil pairing, and its efficient calculation. J. Cryptol. 17(4), 235–261 (2004)
37. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundamentals E85–A(2), 481–484 (2002)
38. Naehrig, M., Niederhagen, R., Schwabe, P.: New software speed records for cryptographic pairings. In: Abdalla, M., Barreto, P. (eds.) LATINCRYPT 2010. pp. 109–123. No. 6212 in LNCS, Springer (2010)
39. Ronan, R., Ó hÉigeartaigh, C., Murphy, C., Scott, M., Kerins, T.: Hardware acceleration of the Tate pairing on a genus 2 hyperelliptic curve. J. Syst. Architect. 53, 85–98 (2007)
40. Rubin, K., Silverberg, A.: Using Abelian varieties to improve pairing-based cryptography. J. Cryptol. 22(3), 330–364 (2009)
41. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000. pp. 26–28 (2000)
42. Taverne, J., Faz-Hernández, A., Aranha, D.F., Rodríguez-Henríquez, F., Hankerson, D., López, J.: Speeding scalar multiplication over binary elliptic curves using the new carry-less multiplication instruction. J. Cryptographic Engineering 1(3), 187–199 (2011)
43. Vercauteren, F.: Optimal pairings. IEEE Trans. Inf. Theory 56(1), 455–461 (2010)
44. Verheul, E.R.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. J. Cryptol. 17(4), 277–296 (2004)

On the Joint Security of Encryption and Signature in EMV

Jean Paul Degabriele[1], Anja Lehmann[2], Kenneth G. Paterson[1], Nigel P. Smart[3], and Mario Strefler[4]

[1] Information Security Group, Royal Holloway, University of London
[2] IBM Research – Zurich
[3] Department of Computer Science, University of Bristol
[4] INRIA / ENS / CNRS, Paris

Abstract. We provide an analysis of current and future algorithms for signature and encryption in the EMV standards in the case where a single key-pair is used for both signature and encryption. We give a theoretical attack for EMV's current RSA-based algorithms, showing how access to a partial decryption oracle can be used to forge a signature on a freely chosen message. We show how the attack might be integrated into EMV's CDA protocol flow, enabling an attacker with a wedge device to complete an offline transaction without knowing the cardholder's PIN. Finally, the elliptic curve signature and encryption algorithms that are likely to be adopted in a forthcoming version of the EMV standards are analyzed in the single key-pair setting, and shown to be secure.

1 Introduction

According to the EMV Co website[1],

> *EMV is a global standard for credit and debit payment cards based on chip card technology. As of end-2010, there were more than 1.24 billion EMV compliant chip-based payment cards in use worldwide. EMV chip-based payment cards, also known as smart cards, contain an embedded microprocessor, a type of small computer. The microprocessor chip contains the information needed to use the card for payment, and is protected by various security features.*

The EMV standards [14, 15, 16, 17] are a complex set of documents defining all aspects of the system, many unrelated to cryptography. In particular, they define three main protocols, SDA, DDA and CDA, which offer different levels of card (and card-holder) authentication and transaction authorization. A good overview of the EMV system can be found in [26], for example.

This paper concerns the security of the encryption and signature schemes used in the EMV standards. These are currently based on the RSA primitive, but there are plans to move to elliptic curve based cryptography at some future date, with

[1] http://www.emvco.com/

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 116–135, 2012.

a draft specification for ECC in EMV having been available for several years [13] and specifying the use of EC-DSA for signatures and ECIES for encryption, and a more recent announcement[2] that EC-Schnorr is under consideration. A main motivation for switching to ECC is that the EMV standards have data formats that do not allow public keys to be larger than 1984 bits.

EMV makes use of signatures during card authentication in DDA and for transaction authentication in CDA. The latter gives the terminal an opportunity to verify the transaction's integrity, without needing to go on-line to communicate with the back-end banking infrastructure to obtain such an assurance. As an option, EMV allows public key encryption to be used for PIN encryption, encrypting the PIN entered by the card-holder at a Point-of-Sale (PoS) terminal to protect it as it is transferred from the terminal to the card, where it is decrypted and compared to the PIN stored on the card. In addition, EMV makes extensive use of symmetric key techniques, but these will not concern us here.

The EMV standards allow the same RSA key-pair to be used for both signature and encryption (see [15, Section 7.1]). This brings benefits in terms of reducing the number of certificates needed in the system, which in turn reduces their storage and processing costs. Our understanding is that the same will be permitted in future versions of EMV using ECC. But is this practice secure? This question is what our paper sets out to address.

Joint security of signature and encryption: The topic of *joint security* of signature and encryption schemes has a fairly extensive history. The first paper to study the problem formally seems to have been by Haber and Pinkas [19], who introduced the concept of a *combined public key scheme* where the existing encrypt, decrypt, sign and verify algorithms of a signature and an encryption scheme are preserved, but where the two key generation algorithms are modified to produce a single algorithm. This algorithm still outputs two key-pairs, but the key-pairs are no longer necessarily independent, and may even be identical. Haber and Pinkas also introduced the natural security model for combined public key schemes, where the adversary against the encryption part of the scheme is equipped with a signature oracle in addition to the usual decryption oracle, and where the adversary against the signature part of the scheme is given a decryption oracle in addition to the usual signature oracle. In this setting, we talk about the *joint security* of the combined scheme. Further work on this topic can be found in [6,25] where the special case of combined schemes built from trapdoor permutations (including the RSA trapdoor permutation) was considered. A recent paper [29] revisits this topic, giving constructions that are provably secure in the standard model and a pathological example showing that schemes that are individually secure may become catastrophically insecure when the same key-pair is used for signature and encryption. That paper also notes that textbook RSA is trivially insecure in such a setting, since access to a decryption oracle (which on input c outputs $m = c^d \bmod N$) instantly allows signature forgeries, simply by setting $c = H(m)$. Fortunately, EMV does not use textbook RSA, so this attack does not apply.

[2] See http://www.emvco.com/faq.aspx?id=38

Previous work on EMV security: Coron *et al.* [8] give an existential forgery attack on the ISO/IEC 9796-2 signature standard, which is also used by EMV. They improved on a previous attack by Coron *et al.* [23], itself an extension of an attack by Desmedt and Odlyzko [11] to larger messages. For operational reasons, these attacks do not threaten the security of EMV signatures. Coron *et al.* [7] showed how to factor the RSA modulus using fault attacks on signatures found in the EMV standard. Using about ten signatures, their attack recovers the factors of N in less than a second. More recently, Smart [33] used a ciphertext validity checking oracle that may exist in certain implementations of the EMV standard to recover the PIN from its encrypted version using as few as 30 queries to the oracle.

At the protocol level, Murdoch *et al.* [26] reported an implementation of a "wedge attack" on the EMV protocol. In the attack, the communications between a PoS terminal and a card are interfered with using a wedge device. This device prevents the PIN validation request from reaching the card and returns a "PIN valid" code to the terminal. This leaves the card and terminal with different views of the protocol, but the card's view of the protocol may not be transmitted to the terminal in a form that is interpretable by the terminal, so the modification may go undetected. Murdoch *et al.* explain how this attack could enable an attacker to make use of a stolen card while not knowing the cardholder's PIN. Their attack may work for all three protocol versions SDA, DDA and CDA. It is notable that this class of attack was already anticipated well before the publication of [26], with protections for CDA already existing in, for example, the EMV Common Payment Application (CPA) specification [12].

Our contribution: None of the previous analysis of EMV examines the security consequences of using the same key-pairs (whether RSA- or ECC-based) for both signature and encryption. Our paper does so, with results that are both negative and positive for EMV:

- We present a theoretical attack against EMV's existing RSA-based algorithms which shows how adversarial access to error information that may be generated during decryption can be used to forge a signature on a freely chosen message. We then show how the attack can be integrated into EMV's CDA protocol flow, enabling an attacker with a wedge device to complete an offline transaction without knowing the cardholder's PIN. Note that this attack would still work even if the proposed countermeasures to the attacks of [26] were adopted. However, the attack is unlikely to work in practice because of other factors which we describe in detail in Section 3.
- We provide positive security results in the joint security setting for the ECC-based encryption and signature schemes that are proposed in [13] and that are likely to be adopted in future EMV standards. More specifically, we prove that the ECIES encryption scheme and the EC-Schnorr signature scheme are jointly secure in the Random Oracle Model, and that ECIES and the EC-DSA signature scheme are jointly secure in the Generic Group Model (GGM). Such results are the best that we can currently hope for, given the state of the art in analyzing ECC signature schemes.

Paper organisation: In the next section, we present our RSA signature forgery attack. In Section 3 we explain how this attack might be used in the context of the EMV protocols to forge a transaction signature for the CDA protocol. Section 4 presents our analysis of the joint security of EMV's likely choices for ECC-based algorithms. We conclude in Section 5.

2 An Attack on Combined Signature and Encryption Schemes from EMV

In this section, we show an attack on the combined signature and encryption scheme from EMV in which an adversary, equipped with a partial decryption oracle, is able to forge a signature on a message of his choice. The partial decryption oracle only tells the adversary whether or not the underlying plaintext is correctly formatted, and the attack is therefore closely related to the attack of Bleichenbacher [3] on RSA with PKCS#1 encoding. However the low-level details differ because of the specific format used in EMV encryption. Note that the attack works independent of the particular encoding used when creating signatures. The idea of using a (partial) decryption oracle to forge signatures in the RSA setting was mentioned in passing in [3] and examined in more detail in [24]. In the next section, we examine the applicability of this attack in the context of the EMV protocol.

Description of the attack. According to [15], Table 25, the plaintext to be encrypted using the ICC's RSA public key (e, N) consists of a 1-byte Data header equal to $\texttt{7F}$, followed by an 8-byte PIN block encoding the PIN in a particular format, followed by the 8-byte ICC Unpredictable Number, and then a final $k - 17$ bytes of random padding. Here k is the length of the ICC's RSA public key in bytes, and is referred to as N_{IC} in the EMV standards.

In what follows, we assume that the decryption process fails with an output of invalid if the obtained plaintext does not begin with a byte $\texttt{7F}$, and otherwise outputs valid. The justification for making this assumption in an EMV context is given in more detail in Section 3. To model this, we equip the adversary with an oracle valid$(\cdot)$ that returns either true or false on input a ciphertext c.

Let k be the byte length of N (so $2^{8(k-1)} \leq N < 2^{8k}$). The 4-digit PIN is first encoded as an 8-byte PIN block $P = \texttt{24}||pp||pp||(\texttt{FF})^5$, where $pp||pp$ denotes a BCD encoding of the digits of the PIN. The PIN block is in turn encoded as a plaintext $m = \texttt{7F}||P||U||R$, where U is the 8-byte ICC Unpredictable Number and R is the $k - 17$ bytes of random padding. A ciphertext c is valid if its decryption m starts with $\texttt{7F}$, so for $B = 2^{8(k-1)}$ we have $127B \leq m < 128B$ when c is valid. We write L for the lower bound $127B$, U for the upper bound $128B - 1$. The modulus N has to be larger than this, so $128B \leq N < 256B$.

Let m be the message for which we wish to forge a signature σ, and let $\mu(m)$ denote the encoding that is applied to m before the signing operation, so that $\sigma = \mu(m)^d \bmod N$. For our attack, μ is arbitrary, but of course EMV uses a specific encoding function. Our attack is then presented in Algorithm 1.

Table 1. Number of queries needed to attack different key lengths

key length	prediction	max.	95th p.	90th p.	median	10th p.	5th p.	min.
512	1408 – 1792	1149495	5764	2837	1462	1166	1088	836
768	1920 – 2304	344278	5900	3318	2029	1682	1594	1233
1024	2432 – 2816	868159	10647	4660	2577	2106	2032	1610
1984	4320 – 4704	221440	17385	8524	4639	3855	3650	3216

The attack is divided into three steps: The blinding step is executed once at the start, and is then followed by a series of iterations where each iteration comprises searching for a new valid ciphertext (step 2) and updating the set of solutions accordingly (step 3). The search is itself divided into three cases of which in each iteration only one will be executed. Step 2a is executed only on the first iteration. If it results in more than one interval, step 2b will be executed in order to reduce this set of intervals to just one. Bleichenbacher [3] presents a heuristic argument to show that a single iteration of step 2b will in most cases be enough. Once the set of solutions is reduced to one interval, each iteration of step 2c will attempt to halve this interval until it is narrowed down to a single value.

Complexity. We estimate the number of oracle accesses in a way analogous to Bleichenbacher [3]. The probability $\Pr(A)$ that a randomly chosen integer $0 \leq m < N$ begins with the byte `7F` is bounded by $2^{-8} < \Pr(A) < 2^{-7}$ if we assume that the bitlength of the modulus N is a multiple of 8 (as it is in EMV). Because we assume that the decryption oracle first checks whether the recovered plaintext starts with `7F` and that we can learn the result of this test, the probability that our ciphertext passes this test is $\Pr(P) = \Pr(A)$. Therefore, step 1 needs about $1/\Pr(P) \in [2^7, 2^8]$ decryption queries on average. Step 2a will take the same number of queries on average, as does each execution of step 2b. Step 2c should on average take 2 queries, since we are trying to reduce the interval by half each time. If we assume that step 2b is executed once, and the modulus has $\log N$ bits, this yields an expected number of queries in the range $[3 \cdot 2^7 + 2 \cdot \log N, 3 \cdot 2^8 + 2 \cdot \log N]$.

Experimental Results. In order to experimentally validate our theoretical predictions of the number of oracle queries needed to forge a signature, we implemented Algorithm 1 in maple and ran it 1000 times for each of four different key lengths, 512 bits, 768 bits, 1024 bits and 1984 bits. The encoded message $\mu(m)$ was picked uniformly at random from the interval $[0, N-1]$ for each trial. While this choice does not respect the EMV encoding, the message is always blinded in step 1, so results are not affected. Table 1 gives an overview of the results.

The first thing to notice is the length of the distribution's tail, as evidenced by the large difference between the 95th percentile and the maximum in each row of Table 1. Similar behaviour was hinted at by Bleichenbacher [3] . On the positive side, we note that the median lies nicely within the expected range,

$c \leftarrow \mu(m)$, $s_0 \leftarrow 1$;
while $\neg\text{valid}(cs_0{}^e \bmod N)$ **do** $s_0 \xleftarrow{\$} \mathbb{Z}_N$; // step 1
$c \leftarrow cs_0{}^e \bmod N$;
$M \leftarrow \{[L, U]\}$;
$i \leftarrow 1$;
while $M \neq \{[a, a]\}$ **do**
  if $i = 1$ **then** // step 2a
    $s \leftarrow \lceil \frac{N+L}{U} \rceil$;
    while $\neg\text{valid}(cs^e \bmod N)$ **do** $s \leftarrow s + 1$;
  end
  if $i > 1 \wedge |M| > 1$ **then** // step 2b
    $s \leftarrow s + 1$;
    while $\neg\text{valid}(cs^e \bmod N)$ **do** $s \leftarrow s + 1$;
  end
  if $i > 1 \wedge |M| = 1$ **then** // step 2c
    $\{[a, b]\} \leftarrow M$, $s_{\text{prev}} \leftarrow s$;
    $r \leftarrow \lceil 2\frac{bs_{\text{prev}} - L}{N} \rceil$;
    flag $\leftarrow$ false;
    while $\neg$flag **do**
      if $\lfloor \frac{L+rN}{b} \rfloor < \lfloor \frac{U+rN}{a} \rfloor$ **then**
        $s \leftarrow \lceil \frac{L+rN}{b} \rceil$;
        flag $\leftarrow \text{valid}(cs^e \bmod N)$;
      end
      $r \leftarrow r + 1$;
    end
  end
  $I \leftarrow \emptyset$;
  foreach $[a, b] \in M$ **do** // step 3
    for $r \leftarrow \lceil \frac{as-U}{N} \rceil$ **to** $\lfloor \frac{bs-L}{N} \rfloor$ **do**
      $I \leftarrow I \cup \{[\max(a, \lceil \frac{L+rN}{s} \rceil), \min(b, \lfloor \frac{U+rN}{s} \rfloor)]\}$
    end
  end
  $M \leftarrow I$, $i \leftarrow i + 1$;
end
return $\sigma \leftarrow as_0^{-1} \bmod N$

Algorithm 1: Forging Algorithm

and the 90th and 95th percentile are below twice, respectively four times, the median. This means that the algorithm behaves well in the majority of cases.

3 Application of the Attack to EMV

In this section, we study to what extent the above attack can be realized in the context of the EMV protocols, and what the impact of the attack would be. We begin by studying in more detail how PIN decryption is specified in EMV, and then examine how the forgery attack can be realized in the context of an offline CDA transaction.

Decryption processing: From [15, Section 7.2] and [16, Section 10.5.1], the ICC (integrated circuit card, or chip) carries out a particular sequence of steps when decrypting. In part, the steps are as follows (we preserve numbering from [15]):

6. Use the ICC private key to decrypt the enciphered PIN data.
7. Check that the ICC Unpredictable Number recovered is equal to the ICC Unpredictable Number that was generated by the ICC with the `GET CHALLENGE` command.
8. Check whether the Data Header byte recovered is equal to 7F.
9. Check whether the PIN in the recovered PIN block corresponds with the PIN stored in the ICC.

Each of steps 6-9 above may fail, in which case the ICC returns an error code `6983` or `6984`. There is one exception, which arises as part of step 9. Here, if all the format checks pass, but the PIN in the plaintext does not match the PIN stored on the card, then the ICC PIN try counter is decremented and an error `0x63C`x is returned, where x is the new value of the PIN try counter. If all the checks succeed and the recovered PIN is correct, the card returns `9000`.

In our attack, each decryption attempt is highly likely to result in an error code being returned by the ICC. This is because, in the attack, the probability that the recovered ICC Unpredictable Number matches the ICC's generated value at step 7 is only 2^{-64}. In order for our attack to proceed at all, we have to make the assumption that the attacker can distinguish a failure at step 7 from a failure at step 8. We also have to assume that step 8 is either carried out before step 7, or is performed irrespective of whether step 7 is successful. These conditions might be met depending on the exact details of how the ICC's decryption procedure is implemented. We note that EMV's CPA specification [12], which specifies a "profile" for EMV cards, does provide more detail on how decryption processing should be performed in order to be compliant with that specification. In particular [12, Section 12.7, Requirements 12.30 and 12.31] and the update in [18] make it clear that step 8 is not carried out if step 7 fails, so our attack would not work for CPA-compliant cards.

The fact that each decryption attempt is highly likely to fail means that the attack can proceed without the risk of the card locking because of the PIN try counter reaching its limit. But cards may also keep a separate PIN decryption

failure counter in addition to the PIN try counter – for example, this is an optional feature in the CPA specification [12]. However this counter and its use are not specified anywhere in the base EMV standards. Even if it is implemented for a particular card, its maximum value may be quite large to cater for bad terminal implementations. For example, in the CPA specification, it is a 2-byte counter, potentially allowing as many as 2^{16} failed decryption attempts. So an attacker may be able to make many decryption queries in an attack, and possibly without any limit.

Integration into the EMV CDA protocol: In an offline CDA transaction the transaction terminates with the card producing a summary of the transaction data (called a Transaction Certificate, TC) which is digitally signed by the card. The card's view of the cardholder verification method used during the transaction is normally present in the TC but in a data format that is proprietary to the issuer bank, and hence the terminal may not be able to check it. This weakness is exploited in the attacks of [26], through which an attacker may be able to carry out transactions using a lost or stolen EMV card without knowing its PIN. A natural fix to prevent these attacks for offline CDA transactions is to standardize the data format so that the terminal can verify that the cardholder verification method reported by the card in the TC matches its view of which method was used. Our attack below shows that, even with this fix in place, an attacker may still be able to complete offline CDA transactions without knowing the card's PIN. Before explaining the attack, we first illustrate how an EMV transaction would typically proceed. To simplify matters we focus mainly on the salient events that occur during an offline transaction where both the terminal and the card support CDA.

Offline CDA transaction processing: When a card is inserted into a terminal, the terminal first requests a list of supported applications. The terminal then selects an application from the list and starts a transaction. An EMV transaction progresses over three phases: card authentication, cardholder verification, and transaction authorization. Card authentication starts with a `READ RECORD` command issued by the terminal to retrieve the card details and other data. Included in the records is the card's RSA public key together with a certificate chain linking the card's private key to a card scheme root key known to the terminal. The terminal then requests the card to sign a provided nonce value. The terminal verifies the signature on the nonce through the card's public key which it in turn authenticates via the supplied certificate chain.

Next is cardholder verification where commonly the cardholder is required to enter his PIN through the terminal's keypad. When the card and terminal both support PIN encryption, the PIN will be encrypted at the terminal and transmitted to the card in encrypted form. PIN verification is initiated by the terminal requesting the PIN try counter from the card. This indicates the number of PIN entry attempts left before the card locks. If the PIN is to be encrypted the terminal also issues a `GET CHALLENGE` command to retrieve the 8-byte ICC Unpredictable number to be included in the encryption of the PIN

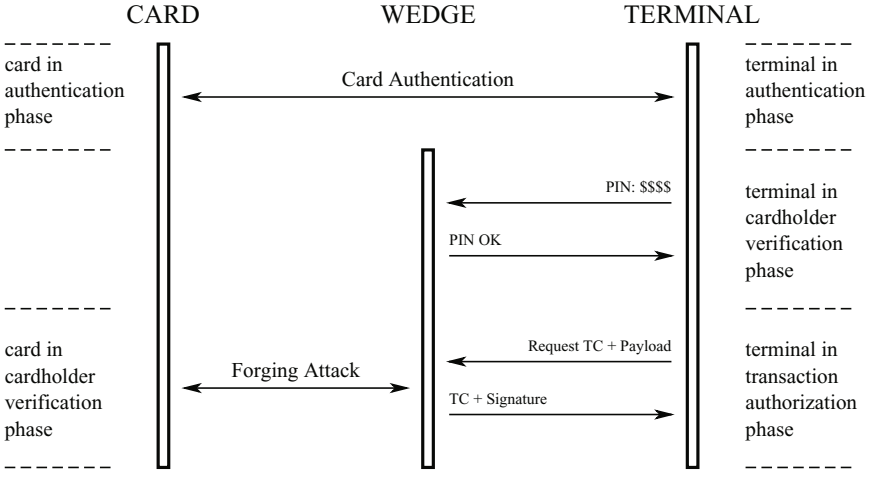


Fig. 1. Executing an offline CDA transaction without the cardholder's PIN

block. The terminal then encrypts the PIN block under the card's public key and submits it to the card for verification through the `VERIFY` command. The card will recover the plaintext using its private key and carry out the decryption checks described above, returning either an error message or the `9000` code that indicates successful PIN verification.

Once the cardholder has been successfully verified, the terminal has to decide whether the transaction requires online authorization from the issuer bank or not. In the latter case, it requests the card to authorize the transaction. A `GENERATE AC` command is sent to the card, containing transaction details and a nonce generated by the terminal. If the card authorizes the transaction, it will respond with a Transaction Certificate (TC) cryptogram. Alternatively it can request the terminal to contact the issuer bank through an Authorisation Request Cryptogram (ARQC), or reject the transaction by responding with an Application Authentication Cryptogram (AAC) which aborts the transaction. The TC contains the transaction details authenticated via a MAC tag (included in the TC) that is computed using a symmetric key which the card and the issuer bank share. The terminal would normally keep a copy of the TC in case there is a dispute, and send a batch of the TCs to the acquiring bank at the end of day for clearing. Note that the MAC tag in the TC can only be verified by the issuer bank. Consequently offline transactions are susceptible to wedge attacks. A CDA card prevents such attacks by additionally providing the terminal with a signature computed over the TC, enabling the terminal to verify the authenticity of the TC offline.

Our offline CDA attack: The attack requires a device that intercepts and manipulates the communication between the card and the terminal. This could for instance be accomplished through a wedge device (a slim device that is inserted

between the card and the terminal) or a fake card which relays communication to the real card as described in [26]. The attack is outlined in Figure 1, showing how the wedge interferes with the normal transaction flow. During the card authentication phase the wedge behaves passively and merely forwards messages between the terminal and the card. Once the terminal initiates the PIN verification phase, the wedge takes over the role of the card. The wedge can return any arbitrary value in response to the `GET CHALLENGE` command. When issued with the `VERIFY` command the wedge will indicate that the PIN verified correctly with a `9000` message, irrespective of the actual PIN value. Thus the attacker can enter any value on the terminal's PIN pad. Since the cardholder verification completed successfully the terminal will go on to request the card to authorize the transaction. At this point the wedge will extract the transaction details and the nonce from the authorization request (`GENERATE AC` command) and compose a TC in the same way that the card would. However, the wedge does not know the card's symmetric key in order to compute the MAC tag that should be present in this TC. Instead, the wedge just assigns a random value to the MAC tag. Since the transaction will be authorized offline, this defect will not be detected until later by the issuer bank.

The wedge now obtains a signature for this TC by mounting the forgery attack of Section 2 against the card. More specifically, the wedge will now impersonate the terminal to the card and initiate a series of PIN verification requests. The card will serve as the validity checking oracle, with the encrypted PIN in the payloads of `VERIFY` commands being replaced by the attack ciphertexts. Prior to each PIN verification request the wedge may need to issue a `GET CHALLENGE` command (see Requirement 12.29 of [12]). Once the wedge has forged a signature over the TC, the wedge forwards the TC together with its signature to the terminal to complete the original transaction.

Impact and practical considerations: In principle, the attack described above enables a wedge device to forge a signature on a TC, which the offline terminal will accept as being valid and thus authorize the transaction. The problem will only come to light later, once the issuer bank tries to verify the MAC in the TC, by which time the attacker equipped with the wedge device and a stolen card, will have made his escape with the purchased goods.

We stress that we have not implemented the above attack. There are several factors that may prevent it in practice. These include the fact that PIN encryption is not yet widely enabled, the fact that cards may use PIN decryption failure counters, the possibility of transaction time-outs being triggered because of the amount of time needed to produce the signature forgery, and the possibility that the `7F` oracle may not be available because of the way in which decryption is implemented (especially for CPA-compliant cards). Nevertheless, the attack illustrates the potential problems that may arise through reusing a keypair in different cryptographic operations.

4 Security Analysis of Combined Encryption and Signature for Elliptic Curve Algorithms

The prior sections detailed possible attacks when re-using the same key for encryption and signature in the existing EMV Co standards. But what can we say about upcoming versions of the standards? EMV Co has indicated that elliptic curve based algorithms are likely to be adopted in future versions of the EMV standards[3]. In particular PIN encryption will be performed by the public key algorithm ECIES [21], whilst digital signatures will be produced using either EC-DSA [20] or EC-Schnorr [22]. Before proceeding it is worth first recapping on what is known about the security of these three algorithms when used on their own, i.e. when used without sharing key-pairs.

ECIES is based on the DHIES encryption scheme [1]. In essence ECIES uses a one-sided static Diffie–Hellman key exchange to obtain a shared secret which is then combined with a one-time IND-CCA secure symmetric encryption scheme via the KEM-DEM paradigm. There are various "options" for use of ECIES in terms of how the agreed Diffie–Hellman key is used to obtain the shared secret. These variants are needed to deal with the well-documented benign malleability of ECIES when used in traditional mode. We use IND-gCCA to denote a scheme which is IND-CCA secure up to benign malleability [2].

The known results for ECIES are that in traditional mode it is IND-gCCA secure in both the random oracle model [1] and in the generic group model [32]. Other variants, as defined in [21], can be shown to be IND-CCA secure. A full description of the various known results can be found in [10].

Security of EC-DSA is more problematic. The only known proof is that it is secure, in the usual EUF-CMA sense, in the generic group model (GGM) [5] with certain requirements on the hash function. No other security proof for EC-DSA is known, and the proof in the GGM makes crucial use of the so-called "conversion" function f which maps elliptic curve points in $E(\mathbb{F}_p)$ to elements of $\mathbb{F}_q$ (note $q \neq p$). We also note that EC-DSA is known to be insecure if used with a hash function for which collisions can be found.

Security of EC-Schnorr is much better understood. It is a classic result of Pointcheval and Stern [30] that Schnorr signatures are secure in the random oracle model, assuming the discrete logarithm problem is hard. In addition EC-Schnorr has been proved secure in the GGM, under the assumption that the hash function used meets further well-defined properties [27]. In particular, even if general collisions can be found in the hash function used in EC-Schnorr, the signature scheme is still secure. Thus EC-Schnorr is resistant to collision attacks on hash functions. Paillier and Vernaud [28] have argued that the above security results in idealized models are probably the best we can hope for, by providing evidence that it is unlikely that a security proof for Schnorr signatures in the standard model will be forthcoming.

In the rest of this section we study the joint security of ECIES and the signature schemes EC-DSA and EC-Schnorr, that is, the security of these schemes

[3] See http://www.emvco.com/specifications.aspx?id=160 for draft specifications

when the same key-pair is used for both signature and encryption. We show that all known security results carry over to the joint setting. Hence, if EMV Co allow the use of a single key-pair for their elliptic curve based algorithms (as they currently do for RSA-based schemes), there are no negative security implications: they would still obtain the strong security guarantees described above, just as if they had used two distinct key pairs.

4.1 Security Models for Joint Security

Security notions for the combinations of a signature scheme and a public key encryption scheme that shares the same keypair (pk, sk) where given in [29]. Such a combined signature and encryption scheme consists of a tuple of algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Enc}, \mathsf{Dec})$ where $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ form a signature scheme and $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ form a PKE scheme. The notions can easily be extended to the case of a hybrid encryption scheme, i.e. a scheme following the KEM-DEM paradigm for constructing a public key encryption scheme. Such a combined signature and KEM scheme is given by a tuple of algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$. To avoid confusion we refer to security of the KEM as KEM-IND-CCA.

In the full version of the paper we present the security notions for combined schemes constructed in the KEM-DEM paradigm. In this shorter version the reader should simply note that the standard notions of EUF-CMA and KEM-IND-CCA security are augmented by giving the adversary additional access to a signature (for KEM-IND-CCA/KEM-IND-gCCA) and decapsulation (for EUF-CMA) oracle that can be queried under the same challenge public key. Informally, we say that a combined scheme is *jointly secure* if it is both EUF-CMA secure in the presence of a decapsulation oracle and KEM-IND-CCA (or KEM-IND-gCCA) secure in the presence of a signing oracle.

4.2 ECIES, EC-Schnorr and EC-DSA in a Nutshell

In this section we briefly describe ECIES, EC-DSA and EC-Schnorr schemes according to their respective ISO specifications.

ECIES Encryption Scheme

ECIES is a public-key encryption scheme based on elliptic curves which follows the KEM-DEM paradigm, i.e., it consists of a key encapsulation scheme ECIES-KEM, and a data encapsulation scheme DEM which uses one-time pad encryption in combination with HMAC [13,21]. Let (E, G, q) be system parameters specifying a secure elliptic curve E together with a point G that generates a secure cyclic subgroup of E, with prime order q, let $\mathsf{KDF} : \{0,1\}^m \to \{0,1\}^*$ be a key-derivation function, $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{\ell_H}$ be a hash function and $\mathsf{HMAC}_{\mathcal{H}}$ be the HMAC-transform of $\mathcal{H}$.

Key Generation. ECIES-$\mathsf{KEM.KGen}$ gets as input the system parameters (E, G, q), chooses a random integer $x \xleftarrow{\$} \{1, \ldots, q-1\}$, sets $Y := xG$ and outputs the keys $pk = (E, G, q, Y)$ and $sk = x$.

Key Encapsulation. ECIES-KEM.Enc on input the public key pk and a message length ℓ_M chooses a random value $r \xleftarrow{\$} \{1, \ldots, q-1\}$, sets $C := rG$ and $K := \mathsf{KDF}([rY]_x)$ where $[rY]_x$ is the x coordinate of rY. The key-derivation function KDF is specified as

$$\mathsf{KDF}(m) = \mathcal{H}(m \| \langle 1 \rangle_{32}) \ \| \ \mathcal{H}(m \| \langle 2 \rangle_{32}) \ \| \ \mathcal{H}(m \| \langle 3 \rangle_{32}) \ \| \ \ldots \mathcal{H}(m \| \langle n \rangle_{32})$$

for $n = (\ell_M + \ell_H/2)/\ell_H$ and where $\langle i \rangle_{32}$ denotes the integer i in binary encoding, represented with 32 bits. It outputs the encapsulated key as (C, K).

Key Decapsulation. ECIES-KEM.Dec on input the public key $sk = x$ and an encapsulated key C first computes $Q := xC$ and checks whether $Q \neq 0$. If so, it outputs $K := \mathsf{KDF}(Q_x)$ where Q_x is the x coordinate of Q and "fail" otherwise.

Data Encapsulation. DEM.Enc on input a message M and a key K first parses the key as $K = K_1 \| K_2$. It then computes $C := M \oplus K_1$ and a tag $t := \mathsf{HMAC}_{\mathcal{H}}(K_2, C)$ and outputs (C, t)

Data Decapsulation. DEM.Dec on input a ciphertext (C, t) and a key K parses the key as $K = K_1 \| K_2$ again. It then computes $M := C \oplus K_1$ and verifies whether $t = \mathsf{HMAC}_{\mathcal{H}}(K_2, C)$. If the verification fails, it returns $\perp$ and M otherwise.

The suggested KEM variant achieves IND-gCCA security only, as it hashes only the x-coordinate of rY instead of the full point which allows for a simple — but in practice harmless – attack against the full-fledged CCA security. Note also that the DEM above is easily malleable if variable-length messages are allowed [31]. However, as the public-key encryption will be applied only to fixed-length messages (containing variable length PINs), the security of the proposed DEM is sufficient.

EC-Schnorr Signature Scheme

Let (E, G, q) be again system parameters specifying a secure elliptic curve E together with a generator point G that generates a secure cyclic subgroup $\mathbb{G}$ with prime order q.

Key Generation. EC-Schnorr.KGen gets as input the system parameters (E, G, q), chooses a random integer $x \xleftarrow{\$} \{1, \ldots, q-1\}$, sets $Y := xG$ and outputs the keys $pk = (E, G, p, Y)$ and $sk = x$.

Signing. EC-Schnorr.Sign on input the secret key $sk = x$ and a message M chooses a random value $r \xleftarrow{\$} \{1, \ldots, q-1\}$, sets $R := rG$ and computes $h := \mathcal{H}(f(R_x) \| f(R_y) \| M)$, where R_x denotes the x-coordinate of R and $f(\cdot)$ is a conversion function that converts a field element into a bit-string. It further computes $s := r + hx \mod q$ and outputs the pair (h, s) unless $s = 0$ or $h = 0$. In this latter case the whole procedure is repeated.

Verification. EC-Schnorr.Verify on input the public key pk, message M and a signature (h, s) first verifies whether $h \neq 0$ and lies in the domain of the hash

function and if $s \in \{1, \ldots, q-1\}$. If one of the conditions is not fulfilled it outputs "invalid", otherwise it continues the verification and computes R' as $R' := sG - hY$ and $h' := \mathcal{H}(f(R'_x) \| f(R'_y) \| M)$. If $h' = h$ it outputs "valid" and "invalid" otherwise.

EC-DSA Signature Scheme

Let (E, G, q) be again system parameters specifying a secure elliptic curve E together with a generator point G that generates a secure cyclic subgroup with prime order q.

Key Generation. EC-DSA.KGen gets as input the system parameters (E, G, q), chooses a random integer $x \xleftarrow{\$} \{1, \ldots, q-1\}$, sets $Y = xG$ and outputs the keys $pk = (E, G, p, Y)$ and $sk = x$.

Signing. EC-DSA.Sign on input the secret key $sk = x$ and a message M chooses a random value $k \xleftarrow{\$} \{1, \ldots, q-1\}$, sets $R := kG$. If $R_x = 0$, where R_x denotes the x-coordinate of R, this step is repeated. Otherwise r is set to be the value of R_x mapped (via it's bit representation) to an element of $\mathbb{F}_q$. The signer then computes $h := \mathcal{H}(M)$ and $s := (h + r \cdot x)/k \mod q$ and outputs the pair (r, s), unless $s = 0$ or $r = 0$. In this latter case the whole procedure is repeated. For future reference the function which sends R to r used in signing is referred to as the "conversion function", we write $r := f(R)$.

Verification. EC-DSA.Verify on input the public key pk, message M and a signature (r, s) first verifies whether r and s lie in the range $\{1, \ldots, q-1\}$. The value $h := \mathcal{H}(M)$ is computed and the verifier computes $a := h/s \mod q$ and $b := r/s \mod q$. Then the value $R' := aG + bY$ is computed, and the signature is accepted if and only if $r = f(R')$.

4.3 On the Joint Security of ECIES and EC-Schnorr

We first sketch why ECIES and EC-Schnorr are jointly secure in the ROM. It is clear we need only show that ECIES-KEM and EC-Schnorr are jointly secure, see the full version for a proof of this elementary result. Thus in the signature game for EC-Schnorr we need to simulate the ECIES decapsulation queries, and in the security game for ECIES-KEM we need to simulate the signing queries for EC-Schnorr. We do not present the proofs in full detail, but simply explain how these extra simulations can be incorporated into the existing proofs.

Security of the KEM Component in the ROM. We start with the simpler case of showing that one can answer EC-Schnorr signature queries within the ECIES-KEM security proof without sacrificing security. Roughly, the ECIES-KEM proof from [9,1] reduces the IND-(g)CCA security to the gap-Diffie-Hellman problem by embedding a DH challenge into the public key and the challenge ciphertext. Recall that the symmetric key in ECIES-KEM is derived from applying the key derivation function KDF on a "Diffie-Hellman key". It is shown that if the KDF

is assumed to be a random oracle, a successful adversary against the ECIES-KEM needs to query the DH-key — and thus the solution to the DH challenge — to the random oracle. In our joint setting we have to reduce this to assuming that the hash function $\mathcal{H}$, that is used to construct the KDF, is a random oracle instead. This stems from the fact that the signature component makes use of a hash function as well and we only want to assume a single random oracle. However, it is easy to see that the KDF as described above inherits the random oracle property from the underlying hash function. We further see that every call to $\mathcal{H}$ is then of the form $P_x \| \langle i \rangle_{32}$, where i is an integer and P_x is a point on the elliptic curve, which allows to adapt the argumentation from the original proof. Note that the DHIES proof in [1] needs to be slightly modified to the case of ECIES due to the benign malleability for the standardized variant of ECIES-KEM mentioned above, but this is a trivial exercise.

To simulate EC-Schnorr queries within this proof we run the "standard" signature simulation from the forking-lemma proof of Schnorr signatures [30]; i.e. the simulator generates h and s at random in $\{1, \ldots, q-1\}$, then defines $R = sG - hY$ and then "patches" the random oracle so that $h = \mathcal{H}(R_x \| R_y \| M)$, where M is the message being signed. One immediately sees that the input to the oracle in this simulation will never interfere with the input to the oracle for the decapsulation queries, and vice-versa. That is, an adversary can not exploit the signature oracle to obtain decryptions of encapsulated keys. Thus the simulation of the signing oracle is perfect, in the random oracle model.

Hence, security of ECIES-KEM in the presence of an EC-Schnorr signing oracle is guaranteed, as long as the conditions for the proof of ECIES-KEM are satisfied. This is that the gap-Diffie–Hellman (gap-DH) problem is hard; i.e. an adversary cannot solve the Diffie–Hellman problem even when given access to a decision Diffie–Hellman (DDH) oracle.

Security of the Signature Component in the ROM. Security of EC-Schnorr in the presence of an ECIES decapsulation oracle follows in much the same way. We simply need to modify the standard forking-lemma based proof of Schnorr signatures so that the adversary in addition has a decapsulation oracle for ECIES-KEM. In the proof the decapsulation queries are simply answered by using the decapsulation simulator from the proof of ECIES-KEM in the random oracle model. Again this latter simulation usually treats KDF as the random oracle, but this is easily replaced by assuming $\mathcal{H}$ is a random oracle instead. Again we also see that the two uses of the random oracle are compatible with each other, due to the sizes of the input values. Thus, we can run the simulations of the hash function for the EC-Schnorr proof and for the ECIES-KEM part in parallel (as the simulator can easily detect if the adversary uses the RO in the context of the signature or the ECIES-KEM component). However, to ensure the hash function queries for the ECIES component are answered correctly the simulator will need access to an oracle which solves DDH. Thus whilst EC-Schnorr is secure assuming the DLP problem is hard, the scheme is only jointly secure assuming DLP is hard even when given access to a DDH oracle. We call this the gap-DLP problem (by analogy with the more standard gap-DH problem mentioned above).

Putting these two informal arguments together, we have:

Theorem 1. *In the random oracle model ECIES-KEM and EC-Schnorr are jointly secure if the gap-DLP problem and gap-DH problem are both hard.*

By adapting the method in the following section one can also show that ECIES-KEM and EC-Schnorr are jointly secure in the GGM if the hash function is random-prefix (second-)preimage resistant and the conversion function f is partially invertible and uniform. This is done by combining the proof in the GGM below for ECIES-KEM with the proof of EC-Schnorr in the GGM found in [27]. We refer to [27] for a definition of what it means for a hash function to be random-prefix (second-)preimage resistant.

4.4 On the Joint Security of ECIES and EC-DSA

We now sketch why ECIES-KEM in combination with EC-DSA are jointly secure in the generic group model (GGM). The idea of the generic group model is that an adversary can not exploit any concrete feature of the group, i.e., the group is considered to be ideal. This is modeled by giving the adversary only oracle — and thus indirect — access to the group elements and group operations. That is, for any input $i \in \mathbb{Z}_q$ the oracle will return the representation $\tau(i) \in \mathbb{G}$, and for any query $(\tau(i), \tau(j), a, b)$ it responds with the element $\tau(ai + bj)$. Note that for the latter query one does not necessarily have to know i or j. To ensure that the oracle is the only way to perform operations on the representations of the group elements, the encoding function τ is chosen randomly from the set of all possible mappings from $\mathbb{Z}_q \to \mathbb{G}$.

Similar to the ROM proof discussion above, we show that one is able to simulate the additional decryption or signing oracle while retaining the original proofs for EC-DSA or ECIES in the GGM. Again, we focus on showing how to handle the extra simulations in the existing proofs.

Security of the KEM Component in the GGM. For full ECIES (i.e. KEM+DEM), security in the GGM was shown under the DDH assumption (which trivially holds in the GGM) and the security of the DEM, but interestingly without any assumptions on the key derivation function. When switching to ECIES-KEM only, it however requires some uniformity property of the KDF as briefly mentioned by Shoup [31]. This property roughly says that the output of a KDF/hash function on a random (and secret) input can not be distinguished from a truly random value.

The proof for ECIES-KEM starts with a tiny game hop, where in the modified KEM-IND-CCA game the challenge ciphertext-key-pair for $b = 1$ is generated as $(rG, \mathsf{KDF}(zG))$ for a random $z \in \{1, \ldots, q-1\}$ instead of (rG, K^+) with K^+ being a random key. Due to the uniformity property of the KDF, this can not result in a noticeable change of $\mathcal{A}$'s success probability. For any successful adversary in the new game that breaks the KEM-IND-CCA property, we can now easily obtain an equally successful adversary breaking the DDH assumption, which can only happen with exponentially small probability in the GGM.

That is on input $(\tau(x), \tau(y), \tau(z))$ an adversary has to output 0 if it believes that $z = xy$ and 1 otherwise. To this end, we embedded the DDH challenge in the public key and challenge ciphertext as $pk = \tau(x)$ and $C^* = (\tau(y), \mathsf{KDF}(\tau(z))$. Thus, if z is a valid DH value, this corresponds to $b = 0$ where the key has the correct form $\mathsf{KDF}(\tau(xy))$ whereas it will correspond to $b = 1$ where a derivation on a random point is given otherwise.

To answer queries to the decryption, signing and group oracle in a consistent way, the simulator maintains a list $\mathcal{L}$ of tuples $(\tau(i), v, w)$ which denotes that the adversary had learned (either directly via a GG call or indirectly via a decryption/signing query) the representation $\tau(i)$ of $i = v + w \cdot x$. For any query $(\tau(i), \tau(j), a, b)$ to the group operation oracle, where $\tau(i), \tau(j)$ must be encodings from previous queries, the simulator first checks if $\mathcal{L}$ contains an entry $(\tau(k), av_i + av_j, bw_i + bw_j)$. If so, it returns $\tau(k)$, otherwise it chooses a random representation $\tau(k) \in \mathbb{G}$ which is different from all other elements stored in $\mathcal{L}$, adds $(\tau(k), av_i + av_j, bw_i + bw_j)$ to its list and returns $\tau(k)$.

The list is further initialized with the tuples $(\tau(1), 1, 0)$ for the generator of the group and $(\tau(x), 0, 1)$ for the public key, where $\tau(1)$ is randomly chosen and $\tau(x)$ is the first value of the DDH challenge.

When responding to a decryption query $\tau(r)$, the simulator first obtains $(\tau(r), v, w)$, from its record and checks if $\mathcal{L}$ already contains an entry $(\tau(k), 0, v + w\bar{x})$. If not it chooses a random representation $\tau(k)$ and adds the tuple $(\tau(k), 0, v + w\bar{x}))$ to $\mathcal{L}$. Note that we can not evaluate $v + wx$ since we do not know x, thus we keep it is as a polynomial with $\bar{x}$ denoting a variable. The decryption oracle finally returns $K \leftarrow \mathsf{KDF}(\tau(k))$.

To simulate signature queries the oracle on input of some message M, chooses a random element $\tau(k)$, computes r, h according to the EC-DSA signing algorithm, chooses $s \in \{1, \ldots, q-1\}$ at random and returns (r, s) to the adversary. It further adds the tuple $(\tau(k), h/s, r/s)$ to $\mathcal{L}$. Thus, when an adversary wants to verify a signature (r, s, M) it must obtain the value $\tau(h/s + rx/s)$ from its group oracle which will then respond with $\tau(k)$ again.

Further, as long as no two entries $(\tau(i), a_i, b_i)$, $(\tau(j), a_j, b_j)$ with $a_i + b_i\bar{x} = a_j + b_j\bar{x}$ but $\tau(i) \neq \tau(j)$ exist, the simulation of the decryption and signature oracle are perfect. The probability of this event can be upper bounded by $|\mathcal{L}|^2/q$.

Security of the Signature Component in the GGM. Brown proved in [4] that EC-DSA is secure in the generic group model if the conversion function f is almost invertible and the hash function is uniform, collision resistant and zero-finder-resistant. The proof mainly shows that each successful forgery (M^*, s^*, r^*) requires an entry $(\tau(k^*), \mathcal{H}(M^*)/s^*, r^*/s^*)$ where $r^* = f(\tau(k^*))$. Depending on the event that triggered this entry, reductions to the underlying hash function assumptions are given. To embed a message for the collision-finder the proof handles queries to the group oracle differently from the proof for IND-CCA security above. Namely, for a query $(\tau(i), \tau(j), a, b)$ where $(\tau(k), a_i + b_i, a_j + b_j)$ is not defined yet, it chooses a random message M and computes $\tau(k) \leftarrow f^{-1}(\mathcal{H}(M) \cdot (a_i + a_j)^{-1} \cdot (b_i + b_j))$. For this step the invertibility of f and the uniformity of h are required, as $\tau(k)$ should not leak information about M.

Queries M to the signing oracle are answered exactly as defined by the EC-DSA algorithm. Here the secret key x is chosen at random, but known to the simulator since we do not play against any computational assumption in the generic group. The simulator further adds for each query an entry $(\tau(k), \mathcal{H}(M)/s, r/s)$ to $\mathcal{L}$. To handle the additional decryption queries, the simulator reacts as above, i.e., on input a ciphertext $\tau(r)$, it obtains $(\tau(r), v, w)$, from its record and checks if $\mathcal{L}$ already contains an entry $(\tau(k), 0, v + wx)$ (recall that this time x is known). If not it chooses a random representation $\tau(k)$ and adds the tuple to $\mathcal{L}$. It returns $K \leftarrow \mathsf{KDF}(\tau(k))$. Due to the knowledge of x both simulations are perfect.

Brown showed that if the entry $(\tau(k^*), \mathcal{H}(M^*)/s^*, r^*/s^*)$ corresponding to the valid forgery was created by the group or signing oracle, this requires that either $\mathcal{H}(M^*) = 0$ or $\mathcal{H}(M^*) = \mathcal{H}(M)$ for one of the messages that was "embedded" in the group oracle responses. In our joint setting we have to consider the additional event that the entry was created by the decryption oracle. As all tuples created by the decryption oracle have a zero as second element that requires $\mathcal{H}(M^*) = 0$ which also contradicts the zero-finder-resistance of $\mathcal{H}$.

Putting both of the above arguments together we obtain:

Theorem 2. *In the generic group model ECIES-KEM and EC-DSA are jointly secure if the DDH problem is hard, the hash function $\mathcal{H}$ is uniform, collision-resistant and zero-finder-resistant and the conversion function f is almost invertible.*

5 Conclusions

Our results on RSA in EMV provide an illustration, should one still be needed, that the deployment of cryptographic algorithms having *ad hoc* designs not supported by formal security analysis can lead to potential and actual security weaknesses. This is especially true when the algorithms are used as components in complex protocols where the possible interactions between algorithms are many and therefore hard to assess.

While the key separation principle dictates using different keys for different cryptographic functions, there are performance benefits that may accrue from reusing keys in constrained environments, and the EMV standards allow key reuse for this reason. We have provided positive security results for the likely candidates for ECC-based algorithms in EMV when keys are re-used. Our results rule out large classes of attack, including attacks like those we exhibited for the existing RSA algorithms.

Acknowledgements. All authors would like to acknowledge the partial support by the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II. The first author was also supported by Vodafone Group Services Limited, a Thomas Holloway Research Studentship, and the Strategic Educational Pathways Scholarship Scheme (Malta), part-financed by the European Union – European Social Fund. The third author was also supported by EPSRC Leadership Fellowship, EP/H005455/1. The fourth author

was also supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, and in part by a Royal Society Wolfson Merit Award. The fifth author was also supported in part by the French ANR-07-TCOM-013-04 PACE Project.

We thank the EMVCo Security Working Group for reviewing an earlier version of this paper.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. An, J.H., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
3. Bleichenbacher, D.: Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 1. Springer, Heidelberg (1998)
4. Brown, D.: Generic groups, collision resistance, and ECDSA. Des. Codes Cryptography 35, 119–152 (2005)
5. Brown, D.: On the provable security of ECDSA. In: Seroussi, G., Blake, I.F., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography, pp. 21–40. Cambridge University Press (2005)
6. Coron, J.-S., Joye, M., Naccache, D., Paillier, P.: Universal Padding Schemes for RSA. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 226–241. Springer, Heidelberg (2002)
7. Coron, J.-S., Naccache, D., Tibouchi, M.: Fault Attacks against EMV Signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
8. Coron, J.-S., Naccache, D., Tibouchi, M., Weinmann, R.-P.: Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 428–444. Springer, Heidelberg (2009)
9. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33, 167–226 (2003)
10. Dent, A.W.: Proofs of security for ECIES. In: Seroussi, G., Blake, I.F., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography, pp. 41–66. Cambridge University Press (2005)
11. Desmedt, Y., Odlyzko, A.M.: A Chosen Text Attack on the RSA Cryptosystem and some Discrete Logarithm Schemes. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 516–522. Springer, Heidelberg (1986)
12. EMV Co. EMV Common Payment Application Specification – Version 1.0 (December 2005)
13. EMV Co. EMV Book 2 – Security and Key Management – Version 4.1z ECC – With support for Elliptic Curve Cryptography (May 2007)
14. EMV Co. EMV Book 1 – Application Independent ICC to Terminal Interface Requirements – Version 4.2 (June 2008)
15. EMV Co. EMV Book 2 – Security and Key Management – Version 4.2 (June 2008)
16. EMV Co. EMV Book 3 – Application Specification – Version 4.2 (June 2008)

17. EMV Co. EMV Book 4 – Cardholder, Attendant, and Acquirer Interface Requirements – Version 4.2 (June 2008)
18. EMV Co. EMV Specification Bulletin No. 84 (December 2010)
19. Haber, S., Pinkas, B.: Securely combining public-key cryptosystems. In: ACM Conference on Computer and Communications Security, pp. 215–224 (2001)
20. ISO/IEC. ISO/IEC 14888-3:2006, Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms (2006)
21. ISO/IEC. ISO/IEC 18033-2, Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers (2006)
22. ISO/IEC. Final Draft of ISO/IEC 14888-3:2006, Information technology – Security techniques – Digital signatures with appendix Part 3: Discrete logarithm based mechanisms Amendment 1: Elliptic Curve Russian Digital Signature Algorithm, Schnorr Digital Signature Algorithm, Elliptic Curve Schnorr Digital Signature Algorithm, and Elliptic Curve Full Schnorr Digital Signature Algorithm (2010)
23. Naccache, D., Coron, J.-S., Stern, J.P.: On the Security of RSA Padding. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 1–18. Springer, Heidelberg (1999)
24. Klíma, V., Rosa, T.: Further Results and Considerations on Side Channel Attacks on RSA. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 244–259. Springer, Heidelberg (2003)
25. Komano, Y., Ohta, K.: Efficient Universal Padding Techniques for Multiplicative Trapdoor One-Way Permutation. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 366–382. Springer, Heidelberg (2003)
26. Murdoch, S.J., Drimer, S., Anderson, R., Bond, M.: Chip and PIN is broken. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 433–446 (May 2010)
27. Neven, G., Smart, N.P., Warinschi, B.: Hash function requirements for Schnorr signatures. J. Mathematical Cryptology 3, 69–87 (2009)
28. 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)
29. Paterson, K.G., Schuldt, J.C.N., Stam, M., Thomson, S.: On the Joint Security of Encryption and Signature, Revisited. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 161–178. Springer, Heidelberg (2011)
30. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)
31. Shoup, V.: A proposal for an ISO standard for public key encryption (version 2.1) (2001), http://www.shoup.net/papers/iso-2_1.pdf
32. Smart, N.P.: The Exact Security of ECIES in the Generic Group Model. In: Honary, B. (ed.) IMACC 2001. LNCS, vol. 2260, pp. 73–84. Springer, Heidelberg (2001)
33. Smart, N.P.: Errors Matter: Breaking RSA-Based PIN Encryption with Thirty Ciphertext Validity Queries. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 15–25. Springer, Heidelberg (2010)

New Constructions of Efficient Simulation-Sound Commitments Using Encryption and Their Applications

Eiichiro Fujisaki

NTT Information Sharing Platform Laboratories, NTT Corporation,
3-9-11 Midori-cho Musashino-shi, Tokyo 180-8585 Japan

Abstract. Simulation-sound trap-door commitment (SSTC) schemes are an essential ingredient for making non-malleable and universally composable protocols. In previous work, the SSTC schemes and their variants are all constructed in the same framework based on digital signatures. In this paper, we provide new constructions of SSTC schemes using *encryption*, which is somewhat surprising, because of the tight relationship between SSTC and digital signature schemes. Although our constructions require a few rounds of interactions between a committer and a receiver and the notion of public-key encryption could be stronger than digital signature, the resulting instantiations are much more efficient than these based on digital signature schemes. In particular, we present an efficient SSTC scheme under the CDH assumption in the bilinear groups, with a tight security reduction and short public key parameters, and the first efficient SSTC scheme under the factoring assumption. Our interactive SSTC schemes inherit properties of the non-interactive version of SSTC schemes to construct non-malleable and universally composable protocols.

1 Introduction

The notion of commitment is central in cryptographic protocol design. A commitment scheme is a two-phase protocol between two probabilistic polynomial-time interactive algorithms, committer C and receiver R. In the committing phase, C takes message m as input and commits to message m by interacting with R, and in the opening phase, C opens m with some witness. A commitment scheme is required to have the *binding* property, meaning that once the commitment phase is completed, C cannot open the commitment to two different messages, except with a negligible probability, and the *hiding* property, meaning that for any $m \neq m'$ (of the same length), a commitment to m (i.e., a view of R in interaction with C on m in the committing phase), is indistinguishable from a commitment to m'. A commitment scheme comes in two different favors, *statistically binding* and *statistically hiding*. In statistically-binding commitments, the binding property holds against unbounded adversaries, whereas in statistically-hiding commitments, the hiding property holds against unbounded adversaries. By construction, a commitment scheme never holds statistically binding and statistically hiding at the same time.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 136–155, 2012.

1.1 Simulation-Sound Commitments

A trap-door commitment is a commitment scheme with an additional *equivocability* property. In such a commitment scheme, there is trap-door information that would allow one to open a commitment in any possible way. A trap-door commitment scheme remains (computational) binding without the knowledge of trap-door information. A simulation-sound trap-door commitment (SSTC) scheme [14,8,21] is a trap-door commitment scheme with a strengthened binding property, called *simulation-sound binding*. Roughly speaking, in an SSTC scheme, an adversary cannot equivocate on a commitment even *after* seeing the equivocation of an unbounded number of different commitments.

The definition of SSTC looks somewhat artificial, but the notion of SSTC implies the notion of *reusable non-malleable commitment with respect to opening* [21,16]. A SSTC scheme can compile any Σ-protocol into a *left-concurrently* non-malleable zero-knowledge proof of knowledge protocol [14,21,15]. Similarly, an SSTC scheme can compile any Ω-protocol into a *left and right-concurrently* (or fully) non-malleable zero-knowledge proof of knowledge protocol (and with some extra modification even universally composable for static corruptions) [14,21]. By combining with *mixed* commitments [9], an SSTC scheme can yield a universally composable (interactive) *string* commitment scheme for at least static corruptions with constant size common reference strings (see Chapter 6 in [16]), where the constant size CRS means that the length of CRS is independent of the number of players. Mixed commitment schemes can be *efficiently* constructed from Paillier [24] and its variant [7]. When combined with an SSTC scheme, a mixed commitment scheme based on [24,7] is transformed to an efficient universally composable *string* commitment scheme with a constant size CRS, with a constant expansion factor, which means that when committing to k bits, the communication costs just $O(k)$ bit.

Multi-trapdoor commitments [15] have been proposed independently, but conceptually similar to SSTCs, and what they can achieve is similar. SSTCs are also defined slightly differently [14,8,21,16]. This paper basically follows the definition of [21].

In the literature of SSTC (including its variant, multi-trapdoor commitments) [14,8,15,21,16,10,22], they are all constructed in the same framework based on digital signature schemes. [21] mentioned: *"Interestingly, all of our constructions are heavily based on signature schemes that are existentially unforgeable against adaptive chosen message attacks. We show that this is not a coincidence, in that there is a straightforward conversion of any* SSTC *scheme into a signature scheme."*

All known SSTC schemes (including ours) is defined in the common reference string model. It is not known to be able to achieve SSTC schemes in the plain model.

1.2 Our Results

We present new constructions of SSTC schemes *using encryption*, which is somewhat surprising because of the tight relationship between SSTC and digital

signature schemes. Initially, SSTC schemes are defined in a non-interactive way, but it is somewhat straightforward to extend them into interactive ones. We define SSTC schemes such that the commitment phase is interactive while the opening phase remains non-interactive, which suffices for our purpose. We start by showing a generic construction of a concurrent man-in-the-middle secure (cMiM) identification (ID) scheme from a certain class of public-key encryptions. We then convert an ID scheme so obtained to an (interactive) SSTC scheme. According to the types of underlying encryption schemes, we present generic constructions of two-round or five-round SSTC schemes. The resulting instantiations can be much more efficient than those compiled from digital signature schemes because we have very efficient public-key encryption schemes (of the required class) from weak and concrete number-theoretic assumptions. Indeed, we present an efficient SSTC scheme under the CDH assumption in the bilinear groups, with tight security reduction and short public key parameters, and the first efficient SSTC scheme under the factoring assumption. The interactive version of SSTCs inherits many properties of the non-interactive counterpart for constructing non-malleable and universally composable protocols.

2 Preliminaries

We let $\mathbb{N}$ denote the natural numbers. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. For positive functions, $f = f(n)$ and $g = g(n)$, we say that $f = O(g)$ if $\lim_{n\to\infty} f(n)/g(n) = c$ for some fixed constant c, and $f = \omega(g)$ if $\lim_{n\to\infty} g(n)/f(n) = 0$. We let $\mathsf{negl}(n)$ denote an unspecified function $f(n)$ such that $f(n) = n^{-\omega(1)}$. Let $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ be probability ensembles such that each X_n and Y_n ranges over $\{0,1\}^n$. We write $X \approx_s Y$ to denote X and Y are statistically indistinguishable. We say that X and Y are computationally indistinguishable if for every polynomial-size circuit family $D = (D_n)_{n>0}$ (ranging over $\{0,1\}$), $\{D_n(X_n)\}_{n\in\mathbb{N}} \approx_s \{D(Y_n)\}_{n\in\mathbb{N}}$. We write $X \approx Y$ to denote that X and Y are computationally or statistically indistinguishable.

2.1 Simulation-Sound Trap-Door Commitment Scheme

We consider commitment schemes equipped with *tags*, i.e., tag-based commitment schemes. We define such schemes that the commitment phase is *interactive* and the opening phase is *non-interactive*. In addition, we focus only on schemes defined in *the common reference string model*. This setting fits our purpose.

Tag-Based Commitment Scheme: A tag-based commitment scheme $\mathsf{CS} = (\mathsf{KGen}, \mathsf{Cc}, \mathsf{Rc}, \mathsf{Rv})$ consists of the following algorithms: KGen is a probabilistic polynomial-time (PPT) algorithm that on input 1^n outputs pk, which seen as the common reference string for the other algorithms. $\langle \mathsf{Cc}(m), \mathsf{Rc}\rangle(pk, \mathsf{tag})$ is an interactive protocol between two interactive algorithms, Cc and Rc, where Cc takes as input, pk, tag $\mathsf{tag} \in \{0,1\}^n$, and message $m \in \{0,1\}^n$ and Rc takes as input pk and tag. After the interaction, Cc outputs dec while Rc outputs Rc's view view, consisting of the transcript of the communication with Cc and

the local input to Rc (including random coins to Rc). We write $(\mathsf{dec}, \mathsf{view}, b) \leftarrow \langle \mathsf{Cc}(m), \mathsf{Rc} \rangle (pk, \mathsf{tag})$ to denote the experiment of one execution above, where $b = 1$ if Rc accepts this communication; otherwise, $b = 0$. $\mathsf{Rv}(pk, \mathsf{tag}, \mathsf{view}, m, \mathsf{dec})$ is a deterministic algorithm that takes $(pk, \mathsf{tag}, \mathsf{view}, m, \mathsf{dec})$ and outputs a bit b'. We require that CS satisfies the following properties:

(Completeness). For every $\mathsf{tag} \in \{0,1\}^n$ and every $m \in \{0,1\}^n$, $\Pr[pk \leftarrow \mathsf{KGen}(1^n); (\mathsf{dec}, \mathsf{view}, b) \leftarrow \langle \mathsf{Cc}(m), \mathsf{Rc} \rangle (pk, \mathsf{tag}): b = 1 \wedge \mathsf{Rv}(pk, \mathsf{tag}, \mathsf{view}, m, \mathsf{dec}) \neq 1 \,] = \mathsf{negl}(n)$.

(Binding). For every non-uniform PPT $\mathsf{Cc}^* = (\mathsf{Cc}_1^*, \mathsf{Cc}_2^*)$, and every $\mathsf{tag} \in \{0,1\}^n$,

$$\Pr\left[\begin{array}{l} pk \leftarrow \mathsf{KGen}(1^n);\ \mathsf{tag} \leftarrow \mathsf{Cc}_1^*(pk);\ (m_1, m_2, \mathsf{dec}_1, \mathsf{dec}_2, \mathsf{view}, b) \leftarrow \langle \mathsf{Cc}_2^*, \mathsf{Rc} \rangle (pk, \mathsf{tag}) : \\ \mathsf{Rv}(pk, \mathsf{tag}, \mathsf{view}, m_1, \mathsf{dec}_1) = \mathsf{Rv}(pk, \mathsf{tag}, \mathsf{view}, m_2, \mathsf{dec}_2) = 1 \wedge (m_1 \neq m_2) \wedge b = 1 \end{array}\right]$$

$= \mathsf{negl}(n)$, where $(m_1, m_2, \mathsf{dec}_1, \mathsf{dec}_2)$ denotes the output of Cc_2^*, view is Rc's view.

(Hiding). For every non-uniform PPT Rc^*, every tag, $m_1, m_2 \in \{0,1\}^n$, where $m_1 \neq m_2$, two Rc^*'s view are indistinguishable, i.e., $\{\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m_1, n)\} \approx \{\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m_2, n)\}$, where $\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m_b, n)$ is: $pk \leftarrow \mathsf{KGen}(1^n)$; $(\mathsf{dec}_b, \mathsf{view}_b, b') \leftarrow \langle \mathsf{Cc}(m_b), \mathsf{Rc}^* \rangle (pk, \mathsf{tag})$; return view_b. We require that even if Rc^* aborts the communication, its view on m_1 should be indistinguishable from the view on m_2. We say that Cc is statistically hiding if $\{\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m_1, n)\} \approx_s \{\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m_2, n)\}$.

Trap-Door Commitment Scheme: A tag-based trap-door commitment scheme is a tag-based commitment scheme with an additional *equivocal* property so that trap-door key tk for each pk can open a commitment in any possible way. Formally, a tag-based trap-door commitment scheme $\mathsf{TC} = (\mathsf{KGen}, \mathsf{Cc}, \mathsf{Rc}, \mathsf{Rv}, \mathsf{TKGen}, \mathsf{TCc}, \mathsf{TDc})$ consists of the following algorithms: $(\mathsf{KGen}, \mathsf{Cc}, \mathsf{Rc}, \mathsf{Rv})$ is a tag-based commitment scheme. TKGen is a PPT algorithm that takes 1^n and outputs (pk, tk). TCc is a probabilistic interactive algorithm that takes as input (pk, tk) generated by $\mathsf{TKGen}(1^n)$ and tag $\mathsf{tag} \in \{0,1\}^n$. It interacts with Rc and executes $\langle \mathsf{TCc}(tk), \mathsf{Rc} \rangle (pk, \mathsf{tag})$. It finally outputs ξ. We write $(\xi, \mathsf{view}, b) \leftarrow \langle \mathsf{TCc}(tk), \mathsf{Rc} \rangle (pk, \mathsf{tag})$ to denote the experience of one execution above, where view denotes Rc's view and $b = 1$ if and only if Rc accepts the communication. TDc is a deterministic algorithm that takes as input (pk, tk), ξ, tag, m, and outputs dec.

(Trap-door property). We require that $\{\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m, n)\} \approx \{\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{TCc}}(m, n)\}$ for every non-uniform PPT Rc^*, every $\mathsf{tag} \in \{0,1\}^n$, and every $m \in \{0,1\}^n$, where $\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{Cc}}(m, n)$ is: $pk \leftarrow \mathsf{KGen}(1^n)$; $(\mathsf{dec}, \mathsf{view}, b') \leftarrow \langle \mathsf{Cc}(m), \mathsf{Rc}^* \rangle (pk, \mathsf{tag})$; return $(\mathsf{view}, m, \mathsf{dec})$, and $\mathsf{view}_{\mathsf{Rc}^*}^{\mathsf{TCc}}(m, n)$ is: $(pk, tk) \leftarrow \mathsf{TKGen}(1^n)$; $(\xi, \mathsf{view}, b') \leftarrow \langle \mathsf{TCc}(tk), \mathsf{Rc}^* \rangle (pk, \mathsf{tag})$; $\mathsf{dec} \leftarrow \mathsf{TDc}(pk, tk, \xi, \mathsf{tag}, m, b')$; return $(\mathsf{view}, m, \mathsf{dec})$. Here we assume that Cc and TDc set $\mathsf{dec} = \perp$ if Rc^* does not accept the communications, i.e., $b' = 0$.

Simulation-Sound Commitment Scheme: A tag-based trap-door commitment scheme $\mathsf{TC} = (\mathsf{KGen}, \mathsf{Cc}, \mathsf{Rc}, \mathsf{Rv}, \mathsf{TKGen}, \mathsf{TCc}, \mathsf{TDc})$ is a tag-based SSTC scheme if the following *simulation-sound binding* property additionally holds.

(Simulation-Sound Binding). For every adversely polynomial-size circuit ensemble A, $\mathsf{Adv}^{\mathsf{ss-bind}}_{A,\mathsf{SSTC}}(n) = \mathsf{negl}(n)$, where $\mathsf{Adv}^{\mathsf{ss-bind}}_{A,\mathsf{SSTC}}(n) \triangleq$

$$\Pr\left[\begin{array}{l}(pk, tk) \leftarrow \mathsf{TKGen}(1^n);\ (\mathsf{tag}, m_1, m_2, \mathsf{dec}_1, \mathsf{dec}_2, \mathsf{view}) \leftarrow A^{\mathsf{TCc}_{tk}, \mathsf{TDc}_{tk}, \mathsf{Rc}}(pk) : \\ \mathsf{Rv}(pk, \mathsf{tag}, \mathsf{view}, m_1, \mathsf{dec}_1) = \mathsf{Rv}(pk, \mathsf{tag}, \mathsf{view}, m_2, \mathsf{dec}_2) = 1 \ \wedge\ (m_1 \neq m_2) \ \wedge\ \mathsf{tag} \notin Q\end{array}\right],$$

where $A^{\mathsf{TCc}_{tk}, \mathsf{TDc}_{tk}, \mathsf{Rc}}(pk)$ is the following left-concurrent man-in-the middle execution: A may activate unbounded polynomial copies of $(\mathsf{TCc}_{tk}, \mathsf{TDc}_{tk})$, while it may activate Rc only once. This execution is done in the manner of the man-in-the-middle execution and A can activate other algorithms with its chosen tags. More precisely, A activates TCc_{tk}, TDc_{tk}, and Rc as follows.

- TCc_{tk}: On input tag, execute $\langle \mathsf{TCc}(tk), A\rangle(pk, \mathsf{tag})$, add tag to Q if A accepts the communication, where set Q is initially empty, and hand (tag, ξ) to TDc_{tk}.
- TDc_{tk}: On input (tag, m), if tag is stored, return $\mathsf{dec} \leftarrow \mathsf{TDc}(pk, tk, \xi, \mathsf{tag}, m)$; otherwise, $\perp$.
- Rc: On input tag, execute $\langle A, \mathsf{Rc}\rangle(pk, \mathsf{tag})$ and output view.

Our definition of SSTC is a natural extention of that of [21] to an interactive one.

2.2 (Tag-Based) Key Encapsulation Mechanisms

A Tag-KEM $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a tag-based KEM [26,1] that consists of three polynomial-time algorithms: Gen, the key-generation algorithm, is a PPT algorithm which on input 1^n outputs a pair of the public and secret keys, (pk, sk). Enc, the encryption algorithm, is a PPT algorithm that takes public key pk and a string $\mathsf{tag} \in \{0,1\}^*$, and produces $(C, K) \leftarrow \mathsf{Enc}(pk, \mathsf{tag}; r)$, picking up random coins r, where $K \in \{0,1\}^n$. Dec, the decryption algorithm, is a deterministic polynomial-time algorithm that takes a secret key sk, tag, and a ciphertext $C \in \{0,1\}^*$, and outputs $\mathsf{Dec}(sk, \mathsf{tag}, C)$. We require that for every (sufficiently large) $k \in \mathbb{N}$, every $\mathsf{tag} \in \{0,1\}^*$ every (pk, sk) generated by $\mathsf{Gen}(1^k)$ and every (C, K) generated by $\mathsf{Enc}(pk, \mathsf{tag})$, it always holds $\mathsf{Dec}(sk, , \mathsf{tag}, C) = K$. A tag is simply a binary string of appropriate length and does not need to have any particular internal structure. Tag-KEM is a generalization of KEM. If the tag is a fixed string, it is a KEM. For using in later sessions, we define the following two binary relations for Tag-KEM Π:

- $\mathbb{R}^{\mathsf{pub}}_{pk,\mathsf{tag}} \triangleq \{(C, r) \,|\, (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$, and
- $\mathbb{R}^{\mathsf{priv}}_{pk,\mathsf{tag}} \triangleq \{(C, K) \,|\, \exists\, r :\ (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$.

We say that a Tag-KEM Π is *publicly verifiable* if there is an efficient algorithm that takes pk (generated by Gen), $\mathsf{tag} \in \{0,1\}^*$, and $C \in \{0,1\}^*$ and evaluates $C \in L^{\mathsf{pub}}_{\mathbb{R}_{pk,\mathsf{tag}}}$.

OW-stCCA. Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a Tag-KEM. We introduce a security notion of one-way security of Tag-KEM against selective-tag chosen-ciphertext attacks. Let $A = (A_1, A_2)$ be a pair of non-uniform PPT algorithms. We define the advantage of $A = (A_1, A_2)$ for Π against one-wayness against selective-tag chosen ciphertext attacks as $\mathsf{Adv}^{\mathsf{ow-stcca}}_{A,\Pi}(n) =$

$$\Pr\left[\begin{array}{l}(\mathsf{tag}^*, s) \leftarrow A_1(1^n);\ (pk, sk) \leftarrow \mathsf{Gen}(1^n);\ C^* \leftarrow \mathsf{Enc}(pk, \mathsf{tag}^*) : \\ A_2^{\mathsf{Dec}(sk,\cdot,\cdot)}(pk, \mathsf{tag}^*, C^*, s) = \mathsf{Dec}(sk, \mathsf{tag}^*, C^*)\end{array}\right],$$

where, when oracle $\mathsf{Dec}(sk, \cdot, \cdot)$ takes (tag, C) from A_2, it returns $\mathsf{Dec}(sk, \mathsf{tag}, C)$ if $\mathsf{tag} \neq \mathsf{tag}^*$, otherwise $\perp$. Tag-KEM Π is said to be $\mathsf{OW\text{-}stCCA}$ if $\mathsf{Adv}^{\mathsf{ow-stcca}}_{A,\Pi}(n) = \mathsf{negl}(n)$ for every A.

OW-ftCCA. Similarly, we define the notion of one-way security of Tag-KEM against full-tag chosen-ciphertext attacks. Let $A = (A_1, A_2)$ be a pair of non-uniform PPT algorithms. We define the advantage of $A = (A_1, A_2)$ for Π against one-wayness against full-tag chosen ciphertext attacks as $\mathsf{Adv}^{\mathsf{ow-stcca}}_{A,\Pi}(n) =$

$$\Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathsf{Gen}(1^n);\ (\mathsf{tag}^*, s) \leftarrow A_1^{\mathsf{Dec}(sk,\cdot,\cdot)}(pk);\ C^* \leftarrow \mathsf{Enc}(pk, \mathsf{tag}^*) : \\ A_2^{\mathsf{Dec}(sk,\cdot,\cdot)}(\mathsf{tag}^*, C^*, s) = \mathsf{Dec}(sk, \mathsf{tag}^*, C^*) \wedge \mathsf{tag}^* \notin Q\end{array}\right],$$

where, when oracle $\mathsf{Dec}(sk, \cdot, \cdot)$ takes query (tag, C) from A_1 or A_2, it returns $\mathsf{Dec}(sk, \mathsf{tag}, C)$, and adds tag to Q, where set Q is initially empty. Tag-KEM Π is said to be $\mathsf{OW\text{-}ftCCA}$ if $\mathsf{Adv}^{\mathsf{ow-ftcca}}_{A,\Pi}(n) = \mathsf{negl}(n)$ for every A.

Generic Conversion from $\mathsf{OW\text{-}stCCA}$ *to* $\mathsf{OW\text{-}ftCCA}$. As shown in [20], a selective-tag secure Tag-KEM can be converted to a full-tag secure Tag-KEM, using a one-time digital signature that is strong existentially-unforgeable against chosen message attacks (sEUF-CMA). In this case, however, we insist that one-time digital signatures can be replaced with (non-interactive) trap-door commitments or chameleon hash functions (as defined in Appendix A.1). Although they are theoretically stronger primitives than one-time signature schemes, they can be often implemented more efficiently under concrete number-theory based assumptions.

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a $\mathsf{OW\text{-}stCCA}$ Tag-KEM and let $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ be a chameleon hash function. We construct a new Tag-KEM $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Gen': Run $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $pk_{ch} \leftarrow \mathsf{CHGen}(1^n)$. Set $pk' := (pk, pk_{ch})$ and $sk' := (pk, pk_{ch}, sk)$. Output (pk', sk').
- $\mathsf{Enc}'(pk', \mathsf{tag})$: Pick up random $r \leftarrow \mathtt{COIN}$ and set $t := \mathsf{CHEval}(pk_{ch}, \mathsf{tag}; r)$, $(C, K) \leftarrow \mathsf{Enc}(pk, t)$, $C' := (C, r)$, $K' := K$, and outputs (C', K').
- $\mathsf{Dec}'(sk', \mathsf{tag}, C')$: Parse C' as C and r. Compute $t = \mathsf{CHEval}(pk_{ch}, \mathsf{tag}; r)$. Output $\mathsf{Dec}(sk, t, C)$.

Theorem 1. *Π' is a* OW-ftCCA *Tag-KEM if Π is a* OW-stCCA *Tag-KEM and $\mathcal{CH}$ is a chameleon hash function.*

Proof. Let A be an adversary to break Π' in the OW-ftCCA game. Let S be a simulator using adversary A and break Π in the OW-stCCA game. First, a simulator sets up $(pk_{ch}, tk_{ch}) \leftarrow \mathsf{CHGen}(1^\kappa)$. It then computes $t^* = \mathsf{CHEval}(pk_{ch}, 0^\kappa; r')$ with random r and hands t^* to the challenger as the selected tag. When S receives pk and $C^* = \mathsf{Enc}(pk, t^*)$, S runs A with pk. When A outputs tag^*, S computes r^* such that $t^* = \mathsf{CHEval}(pk_{ch}, \mathsf{tag}^*; r^*)$ by using tk, and feeds $C^* = \mathsf{Enc}(pk, t^*)$ and r^* to A. If A outputs the decryption, S outputs it. It is obvious by construction that the advantage of S is equivalent to that of A. □

2.3 Sigma-Protocol

Let us remind you of Σ-protocols [5]. Let $R = \{(x, w)\}$ be a binary relation (possibly not only an $\mathcal{NP}$ binary relation). A Σ-protocol for a polynomial-time relation R is the following 3-round (public coin) interactive proof system between the prover and the verifier, with some special properties. Let x be a statement to be proven that there is a witness w such that $(x, w) \in R$. x is given to both the prover and the verifier as common input and w is given only to the prover in advance. A Σ-protocol on common input x is executed as follows: The prover picks up random coins r_a, computes a using statement x and witness w, denoted $a = \Sigma_{\mathsf{com}}(x, w; r_a)$, and sends it to the verifier. The verifier picks up a random $\mathsf{challenge}$ element $c \leftarrow_R \Sigma_{\mathsf{ch}}$, where Σ_{ch} is a uniform distribution over a specified set, and sends it to the prover. The prover responds with $z = \Sigma_{\mathsf{ans}}(x, w, r_a, c)$. The verifier returns a bit $b = \Sigma_{\mathsf{vrfy}}(x, a, c, z)$. We say that (a, c, z) is an accepting communication for x if $\Sigma_{vrfy}(x, a, c, z) = 1$. We require that Σ-protocols satisfy the following properties:

(Completeness). For every r_a (in a specified domain) and every $c \in \Sigma_{\mathsf{ch}}$, it always holds that $\Sigma_{\mathsf{vrfy}}(x, \Sigma_{\mathsf{com}}(x, w; r_a), c, \Sigma_{\mathsf{ans}}(x, w, r_a, c)) = 1$.

(Special Soundness). For every $x \notin L_R$ and every a, there is the only *one* c in Σ_{ch} such that there is z such that $\Sigma_{\mathsf{vrfy}}(x, a, c, z) = 1$. In addition, one can always compute witness w from two accepting communications for x of the form (a, c, z) and (a, c', z'), where $c \neq c'$. The pair of accepting communications, (a, c, z) and (a, c', z'), where $c \neq c'$, is called a *collision*. Note that a collision on x immediately implies that $x \in L_R$.

(Special Honest-Verifier Zero-Knowledge). Given any $x \in L_R$, one can produce a valid transcript $(a, c, z) \leftarrow \mathsf{sim}\Sigma(x, c)$ with the same distribution of real valid transcripts, without knowledge of witness w.

3 Generic Construction of cMiM Secure IDs

We start by observing that any OW-ftCCA Tag-KEM can be converted to a concurrent man-in-the-middle (cMiM) secure tag-based identification (ID) scheme

in a straightforward way[1]. The construction is as follows: Let Π be a OW-ftCCA Tag-KEM. The prover and the verifier have the same tag tag in advance. The verifier simply computes $(C, K) \leftarrow \mathsf{Enc}(pk, tag)$ with public-key pk of the prover and sends ciphertext C to the prover, who returns $K' = \mathsf{Dec}(sk, tag, C)$. The verifier accepts if and only if $K = K'$.

Theorem 2. *The tag-based* ID *scheme shown above is* cMiM *secure if the underlying Tag-KEM is* OW-ftCCA*. In addition, for any non-uniform PPT A against ID under* cMiM *attacks, there is a polynomial-size ensemble S against Tag-KEM Π such that* $\mathsf{Adv}^{\mathsf{cmim}}_{A,ID}(n) \leq \mathsf{Adv}^{\mathsf{ow-ftcca}}_{S,\Pi}(n)$.

Proof. In a cMiM attack for an ID scheme, a cMiM adversary A is allowed only left-concurrency. Therefore, when the cMiM adversary starts only one execution of the ID scheme in the right interaction and sends a tag tag^* to the simulator S, it simply sends it to the encryption oracle $\mathsf{Enc}(pk, \cdot)$ in the OW-ftCCA game, and sends A the challenge ciphertext $C^* \leftarrow \mathsf{Enc}(pk, \mathsf{tag}^*)$. Note that S can simulate any execution of the ID scheme in the left interaction because any tag in the right interaction is not equivalent to tag^* and so S can get any decryption of ciphertexts in the left interaction with the help of the decryption oracle in the OW-ftCCA game. S simply outputs K' that A returns in the right interaction. □

4 Weak Extractable Sigma-Protocol

We introduce a new variant of the Σ-protocol, called the weak extractable Sigma protocols, denoted $\hat{\Sigma}$-protocols. A $\hat{\Sigma}$-protocol $\hat{\Sigma} = (\hat{\mathsf{Gen}}, \hat{\Sigma}_{\mathsf{com}}, \hat{\Sigma}_{\mathsf{ch}}, \hat{\Sigma}_{\mathsf{ans}}, \hat{\Sigma}_{\mathsf{vrfy}}, \mathsf{sim}\hat{\Sigma})$ consists of the following algorithms: $\hat{\mathsf{Gen}}$, the key-generation algorithm, is a probabilistic polynomial-time algorithm which on input 1^k outputs a pair comprising the public and trap-door keys, (pk, tk). Let $R_{pk} = \{(x, w)\}$ be a polynomial-time binary relation indexed by a public-key pk. A $\hat{\Sigma}$-protocol for relation R_{pk} is the following 3-round (public coin) interactive proof system between the prover and the verifier, where letting $(x, w) \in R_{pk}$, x is given to both the prover and the verifier as common input and w is given only to the prover in advance. The prover picks up random coins r_a, computes a using x and w, denoted $a = \hat{\Sigma}_{\mathsf{com}}(x, w; r_a)$, and sends it to the verifier. The verifier picks up a random challenge element $c \leftarrow_R \hat{\Sigma}_{\mathsf{ch}}$, where Σ_{ch} is a uniform distribution over a specified set, and sends it to the prover. The prover responds with $z = \hat{\Sigma}_{\mathsf{ans}}(x, w, r_a, c)$. The verifier returns a bit $b = \hat{\Sigma}_{\mathsf{vrfy}}(x, a, c, z)$. We say that (a, c, z) is an accepting communication for x if $\hat{\Sigma}_{vrfy}(x, a, c, z) = 1$. We require that the $\hat{\Sigma}$-protocol has the following properties:

[1] The idea to use public-key encryption in the concurrent attacks appears in [3], but the paper only starts with IND-CCA2 encryptions. Independently of us, Anada and Arita [2] have proposed the similar idea to ours to construct cMiM IDs.

(Completeness). For every r_a (in a specified domain) and every $c \in \hat{\Sigma}_{\mathsf{ch}}$, it always holds that $\hat{\Sigma}_{\mathsf{vrfy}}\ (x,\ \hat{\Sigma}_{\mathsf{com}}(x, w; r_a),\ c,\ \hat{\Sigma}_{\mathsf{ans}}(x, w, r_a, c)) = 1$.

(Weak Special Soundness). For every $pk \in \hat{\mathsf{Gen}}(1^n)$, every $x \notin L_{R_{pk}}$ and every $a \in \Sigma_{\mathsf{com}}(x, w)$, there is the only *one* c in $\hat{\Sigma}_{\mathsf{ch}}$ such that there is z such that $\hat{\Sigma}_{\mathsf{vrfy}}(x, a, c, z) = 1$. It implies that if there is a collision on x, i.e. a pair of accepting communications for x, (a, c, z) and (a, c', z') where $c \neq c'$, then $x \in L_{R_{pk}}$.

(Weak Extractability). For every $pk \in \hat{\mathsf{Gen}}(1^n)$, every $x \notin L_{R_{pk}}$, every $a \in \hat{\Sigma}_{\mathsf{com}}(x, w)$, and every $c \in \hat{\Sigma}_{\mathsf{ch}}$, one can efficiently check whether there exists z such that $\hat{\Sigma}_{\mathsf{vrfy}}\ (pk,\ x,\ a,\ c,\ z) = 1$, with his knowledge of trap-door key tk.

(Special Honest-Verifier Zero-Knowledge). For every $pk \in \hat{\mathsf{Gen}}(1^n)$ and every $x \in L_{R_{pk}}$, one can produce the valid transcript $(a, c, z) \leftarrow \mathsf{sim}\hat{\Sigma}(x, c)$ with the same distribution of real valid transcripts, without knowing the witness w.

We note that when $x \notin L_R$, the first message sent by the prover commits to c in the statistically binding manner, because there is only one valid c in $\hat{\Sigma}_{\mathsf{ch}}$ due to the weak special soundness property. The weak extractability implies that if trap-door key tk is given and challenge c is given, one can test whether such a first message really commits to c .

5 The SSTC Schemes

We present two constructions of SSTC schemes based on the ID scheme described in Sec. 3. The high-level idea behind our constructions is as follows. We put public-key pk for Tag-KEM $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as the common reference string. In the commitment phase, the receiver creates $(C, K) \leftarrow \mathsf{Enc}(pk, \mathsf{tag})$ and sends C to the committer. Then, the committer simulates the first message, denoted a, of the Σ-protocol on C for relation $\mathbb{R}^{\mathsf{priv}}_{pk,\mathsf{tag}} \triangleq \{(C, K) \,|\, \exists\, r : (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$. Here we note that unlike the prover in the ID scheme above, the committer is not given secret key sk and hence, he does not know $K = \mathsf{Dec}(sk, \mathsf{tag}, C)$. So, the committer is to commit to some c when creating the first message a. Otherwise, we can construct an adversary that decrypts ciphertext C by using the committer who can open the commitment in two ways, which contradicts the one-wayness for Π. On the other hand, if the simulator (playing the role of the committer) is given trap-door key sk, he can decrypt C and create the first message of the Σ-protocol such as $a \leftarrow \Sigma_{\mathsf{com}}(C, K)$. This implies that he can open the commitment a in any possible way, because he can compute z for any message c[2]. Here, we note that even if the simulator shows an adversary two ways of opening of a commitment on a tag, it is still infeasible for the adversary to open a commitment on another tag in two ways, because it implies that it decrypts a ciphertext on a fresh tag, on which the adversary

[2] We note that the idea to use the simulator of a Σ protocol in commitment protocols is not novel. It has their origin in the earlier zero-knowledge papers [12,13]. In addition, all previous SSTC schemes based on digital signatures also use the idea.

never asks for decryption (If it can decrypt the ciphertext, which contradicts OW-ftCCA of Tag-KEM Π). Therefore, this commitment scheme seems to have the simulation-sound binding, too.

This observation above intentionally overlooks an important point. In the commitment phase, an adversary can send *invalid* ciphertexts when he plays the role of the receiver, which ruins the trap-door property of the commitment scheme, because there is no key K such that $K = \mathsf{Dec}(sk, \mathsf{tag}, C)$, and the simulator cannot open a commitment in two ways (or it cannot open a commitment to the value given after the commitment phase was completed). However, if we can make an adversary send only valid ciphertexts, the above observation is true.

Therefore, we now consider two cases. The first case is that Tag-KEM Π is publicly verifiable. As defined in Sec. 2.2, Tag-KEM Π is said to be publicly verifiable if there is an efficient algorithm that is able to check if ciphertext C is *valid*, given only pk, C. For such Tag-KEMs, we can easily fix the scheme above. We just modify the commitment phase so that the committer checks that the ciphertext sent by the receiver is valid. If it is an invalid ciphertext, the committer simply aborts, otherwise he simulates the first message of Σ-Protocol on the valid ciphertext. If Tag-KEM Π is not publicly verifiable, we do not use this strategy, but there is still a case in which we can construct SSTC schemes. In such a case, we require that Π has a weak extractable Σ-protocol for relation $\mathbb{R}^{\mathsf{pub}}_{pk,\mathsf{tag}} = \{(C, r) \,|\, (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$.

Hereafter, we describe a construction of a 2-round SSTC scheme if Tag-KEM Π is publicly verifiable OW-ftCCA, and we provide a construction of a 5-round SSTC scheme if Tag-KEM Π is OW-ftCCA and Π has a weak extractable Σ-protocol for relation $\mathbb{R}^{\mathsf{pub}}_{pk,\mathsf{tag}}$.

5.1 The 2-Round SSTC Scheme from Publicly Verifiable OW-ftCCA Tag-KEM

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a publicly-verifiable OW-ftCCA Tag-KEM. We then construct a 2-round SSTC scheme based on Π as described in Fig. 1.

Common Reference String. (pk, Π), where pk is a public key according to the distribution of $\mathsf{Gen}(1^k)$ in publicly verifiable OW-ftCCA Tag-KEM Π.
The Commitment Phase.
- On given tag (possibly sent by the committer Cc) and given the common reference string (CRS), the receiver Rc computes $(C, K) \leftarrow \mathsf{Enc}(pk, tag)$ and sends C to Cc.
- The committer Cc takes C as well as the CRS, and commits to message m, by computing $(a, z) \leftarrow \mathsf{sim}\Sigma(C, m)$ for relation $\mathbb{R}^{\mathsf{priv}}_{pk,tag} = \{(C, K) \,|\, \exists\, r :\ (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$. Cc sends a to Rc.
- Rc always accepts when it receives a reply from Cc.

The Opening Phase. The committer sends (m, z) to the receiver, who accepts if and only if $\Sigma_{\mathsf{vrfy}}(x, a, m, z) = 1$ for $\mathbb{R}^{\mathsf{priv}}$.

Fig. 1. The 2-round SSTC scheme

Theorem 3. *Let Π be a publicly verifiable* OW-ftCCA *Tag-KEM. The scheme obtained above is an* SSTC *scheme. In particular, for any polynomial-sized ensemble A against* SSTC*, there is a polynomial-sized ensemble S against publicly verifiable Tag-KEM Π such that* $\mathsf{Adv}^{\mathsf{ss-bind}}_{A,\mathsf{SSTC}}(n) \leq \mathsf{Adv}^{\mathsf{ow-stcca}}_{S,\Pi}(n)$.

Proof. (Hiding) Since Π is publicly verifiable, $\{\mathsf{view}^{\mathsf{Cc}}_{\mathsf{Rc}^*}(m,n)\} \equiv \{\mathsf{view}^{\mathsf{Cc}}_{\mathsf{Rc}^*}(m',n)\}$ for every $m = m'$ and every unbounded Rc^*. (Binding) If follows from the difficulty of breaking one-wayness of Π and the special soundness property of Σ-protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$. (Trap-door property) It follows from the special honest-verifiable zero knowledge property of Σ-protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$ and publicly verifiability of Π. (Simulation-sound binding) The proof is similar to the proof of Thm. 2. When an adversary plays the role of the receiver in the commitment phase (i.e., interacting with oracle TCc_{sk}), it may ask TDc_{sk} with (tag, c) and (tag, c'), $c \neq c'$, and receive dec $(= z)$ and dec' $(= z')$, respectively. By special soundness, it leads that the adversary obtains K such that $\mathsf{Dec}(sk, \mathsf{tag}, C) = K$ for some (tag, C) stored in the commitment phase (in the left interaction). The goal of the adversary is to open a commitment in two ways on tag^*, which does not appear in the left interaction. However, opening a commitment in two ways in the right interaction implies that the adversary outputs $K^* = \mathsf{Dec}(sk, \mathsf{tag}^*, C^*)$ for target tag tag^*, which contradicts OW-ftCCA for Tag-KEM Π. □

5.2 The 5-Round SSTC Scheme from OW-ftCCA Tag-KEM

Suppose that Tag-KEM Π is not publicly verifiable, but has a weak extractable Σ-protocol for relation $\mathbb{R}^{\mathsf{pub}}_{pk,\mathsf{tag}}$. In such a case, we can instead construct a 5-round SSTC scheme based on Π if Π is OW-ftCCA .

Since Π is not publicly verifiable, we instead let the receiver prove that he really create a valid ciphertext. To prove this, we let the prover send an IND-CCA2 encryption of challenge $\hat{c}$ to the receiver. Then, we let the receiver produce a Σ protocol for $\mathbb{R}^{\mathsf{pub}}_{pk,\mathsf{tag}}$, as playing the role of the prover, where the committer sends the same $\hat{c}$ as challenge. This is to have a straight-line simulator. However, this does not satisfy soundness, because the CCA encryption conceals the challenge only computationally. So, we need more property in Σ protocol for $\mathbb{R}^{\mathsf{pub}}_{pk,\mathsf{tag}}$, which is called weak extractability.

Let $\Pi' = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an arbitrary standard IND-CCA2 public key encryption where $\mathcal{K}$ is the key generation algorithm, $\mathcal{E}$ is the encryption algorithm, and $\mathcal{D}$ is the decryption algorithm. The construction is shown as in Fig. 2.

Theorem 4. *Let Π be a* OW-ftCCA *Tag-KEM. Let Π' be a* IND-CCA2 *public-key encryption scheme. The scheme obtained above is an* SSTC *scheme:*

*(*Simulation-sound binding*) For any polynomial-sized ensemble A against* SSTC*, there is a polynomial-sized ensembles, S and S', such that* $\mathsf{Adv}^{\mathsf{ss-bind}}_{A,\mathsf{SSTC}}(n) \leq \mathsf{Adv}^{\mathsf{ow-ftcca}}_{S,\Pi}(n)$.

*(*Trap-door property*) In addition,* $\{\mathsf{view}^{\mathsf{TCc}}_{\mathsf{Rc}^*}(m,n)\} \approx_c \{\mathsf{view}^{\mathsf{Cc}}_{\mathsf{Rc}^*}(m,n)\}$*, where the computational distance of the two views is almost twice the advantage of breaking Π' in IND-CCA2 attacks.*

Common Reference String. (pk, pk', Π, Π'), where pk and pk' are each public key according to the distribution of $\mathsf{Gen}(1^k)$ in OW-ftCCA Tag-KEM Π and $\mathcal{K}(1^k)$ in IND-CCA2 Π', respectively.

The Commitment Phase.

- Given tag and given the common reference string (CRS), the committer Cc picks up $\hat{c} \leftarrow_R \hat{\Sigma}_{\mathsf{ch}}$ for $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$ and encrypts it using pk' and randomness r', i.e., $C' \leftarrow \mathcal{E}^{\mathsf{cca}}_{pk'}(\hat{c}; r')$. Cc sends C' to the receiver Rc.
- Given tag (possibly sent by the committer), the CRS, and C' from the committer, the receiver Rc computes $(C, K) \leftarrow \mathsf{Enc}(pk, \mathsf{tag}; r)$. He then computes $\hat{a} \leftarrow \hat{\Sigma}_{\mathsf{com}}(C, r; r_a)$ for $\mathbb{R}^{\mathsf{pub}}_{pk,tag} = \{(C, r) \,|\, (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$, and sends $(C, \hat{a})$ to Cc.
- Cc takes $(C, \hat{a})$, and sends $(\hat{c}, r')$ to Rc.
- Rc aborts if $C' \neq \mathcal{E}^{\mathsf{cca}}_{pk'}(\hat{c}; r')$; otherwise, it replies $\hat{z} \leftarrow \hat{\Sigma}_{\mathsf{ans}}(C, r, r_a, \hat{c})$ for $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$.
- Cc aborts if $\hat{\Sigma}_{\mathsf{vrfy}}(pk, C, \hat{a}, \hat{c}, \hat{z}) \neq 1$ for $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$, otherwise it commits to message m, by computing $(a, z) \leftarrow \mathsf{sim}\Sigma(C, m)$ for relation $\mathbb{R}^{\mathsf{priv}}_{pk,tag} = \{(C, K) \,|\, \exists\, r :\ (C, K) = \mathsf{Enc}(pk, \mathsf{tag}; r)\}$. Cc sends a to Rc.
- Rc always accepts when it gets a reply from Cc.

The Opening Phase. The committer sends (m, z) to the receiver, who accepts if and only if $\Sigma_{\mathsf{vrfy}}(x, a, m, z) = 1$ for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$.

Fig. 2. The 5-round SSTC scheme

Proof. (Simulation-Sound Binding) Let S be an adversary against Π in the OW-ftCCA game, where (tag^*, C^*) denotes the challenge. S sets up $(pk', sk') \leftarrow \mathcal{K}'(1^n)$ by himself and runs A in the simulation-sound binding game. In the right interaction, A can interact with the receiver only once with a tag tag^*, and S can freely decrypt the first message C' sent by A in the right interaction, because he knows sk'. Then, S plays the role of the receiver in the right interaction and sends back C^* with $\hat{a}$, where C^* is the challenge ciphertext in the OW-ftCCA game. $\hat{a}$ is the simulated message such that $(\hat{a}, \hat{z}) \leftarrow \mathsf{sim}\hat{\Sigma}((pk, C^*), \mathcal{D}_{sk'}(C'))$. Since S always knows $\mathcal{D}_{sk'}(C')$, he can always reply with a valid $\hat{z}$. In the left interaction, whenever S receives (tag, C), he can obtain the decryption with the help of the decryption oracle in the OW-ftCCA game, because by definition, $\mathsf{tag} \neq \mathsf{tag}^*$. Therefore, in the left interaction, S can always open a commitment into any message requested by A. Hence, S can perfectly simulate the environment of A in the simulation-sound binding game. When A succeeds to equivocate a commitment in the right interaction, it implies that S can obtain the decryption of C^* on tag^*, because of the special soundness property of the Σ-protocol.

(Trap-door property) Since A receives C' in the left interaction, A has a chance, with some probability, to complete the commitment phase with an invalid C, which makes a difference between Cc and TCc. Namely, TCc always aborts if such

an event occurs, whereas Cc proceeds with the commitment protocol. However, the chance of A can be bounded by the advantage of IND-CCA2 Π'. Consider an IND-CCA2 game of Π', where we construct simulator S as follows: Stakes pk' as input and sets up $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ by itself. it chooses two messages, m and m', and obtains the challenge ciphertext C', which is the encryption of one of the two messages. it then feeds C' to A in the left intersection. Then, A sends back C and $\hat{a}$. Note that if C sent by A is an invalid encryption, $\hat{a}$ must be the statistically binding commitment to m or m' (otherwise, A cannot complete the commitment phase). Since S knows secret key sk and $\hat{a}$ is the first message in a weak extractable Σ-protocol, it can check which of the two messages that A committed to in $\hat{a}$. Therefore, the chance that A can complete the commitment phase with invalid C is bounded by the advantage of IND-CCA2 Π'; otherwise, it leads to a contradiction. □

6 Applications

As with [21], our tag-based SSTC schemes can be converted to body-based SSTC schemes by replacing tags with verification keys *otvk* of a strong one-time signature scheme and in the last message, sending a signature w.r.t. *otvk* on the whole communication of one execution between the committer and the receiver.

We insist that any (interactive or non-interactive) SSTC scheme can be converted to a cMiM secure ID scheme, which can be seen as a generalization of the fact stated in [21] that any non-interactive SSTC scheme can be converted to an EUF-CMA digital signature scheme. Let pk and sk be a common reference string and a trap-door key in an SSTC scheme, respectively. We regard them as public and secret keys for an ID scheme. In order to identify himself, the prover interacts with the verifier, firstly by playing the role of TCc. Then, he shows the verifier that he can open the commitment in two ways. The verifier accepts if the prover succeeds in doing so.

Any non-interactive SSTC scheme can compile any Σ-protocol into a *left-concurrently* non-malleable zero-knowledge proof of knowledge protocol [14,21,15]. By replacing a Σ-protocol with an Ω-protocol, it becomes a *fully concurrently* non-malleable zero-knowledge proof of knowledge protocol [14,21].[3] Our interactive SSTC schemes can safely replace non-interactive ones in these applications without harming non-malleable properties. In addition, by combining with mixed commitments [9], a non-interactive SSTC scheme can yield a universally composable (interactive) *string* commitment scheme [16]. In the model of static corruptions, our interactive SSTC schemes can safely replace non-interactive SSTC schemes. We prove this in the full version. The basic idea is that since corruptions are only static, each party is either corrupted before the

[3] An Ω-protocol is a Σ-protocol in the CRS model with an additional property: Informally, it is such a property that if x is a true instance (namely, $x \in L$) and the Ω-protocol is for language L, then one can directly extract witness w from x and one accepted communication (a, e, z), by using the trap-door information behind the CRS.

protocols start or never corrupted. Therefore, even if we replace non-interactive SSTC commitments with interactive SSTC ones, we do not need to consider corruptions at intermediate states in the commitment phase, and hence universally composability still holds.

Non-interactive SSTC schemes imply reusable non-malleable commitments with respect to opening [21,16]. It is straightforward to show that interactive SSTC schemes also imply reusable non-malleable commitments with respect to opening. A stronger notion is (simulation-based) *concurrent* non-malleable commitment with respect to opening [23]. Our interactive SSTC schemes also imply (simulation-based) concurrent non-malleable commitment with respect to opening. Due to the space limitation, we prove this in the full version, although the proof is almost straight-forward.

7 Instantiations

7.1 2-Round, CDH-Based Implementation

We provide a publicly verifiable OW-ftCCA Tag-KEM based on CDH assumption in the bilinear map, which can be obtained by converting a OW-stCCA Tag-KEM [20,28] with a chameleon hash function. As mentioned in in Sec. 2.2, any OW-stCCA Tag-KEM can be converted to a OW-ftCCA Tag-KEM using a chameleon hash function. We also provide a Σ-protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$.

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear map defined over two (multiplicative) cyclic groups, $\mathbb{G}$ and $\mathbb{G}_T$, of order prime q. By bilinearity, we have $e(x^a, y^b) = e(x, y)^{ab}$ for all $x, y \in \mathbb{G}$, and $a, b \in \mathbb{Z}/q\mathbb{Z}$. Let g be a generator of $\mathbb{G}$ By non-degeneration, $e(g, g)$ is a generator of $\mathbb{G}_T$, too. We assume that the computational Diffie-Hellman (CDH) problem in $\mathbb{G}$ is difficult. Let $\mathcal{H} = \{\mathcal{H}\}_\iota$ be a keyed collision-resistant (CR) hash function family. Let $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ be a chameleon hash function. As a concrete example, we can use a Pedersen commitment with a CR hash function. Let $H, H' : \{0,1\}^* \to \{0,1\}^k$ be CR hash functions. Let $\mathbb{G}'$ be a cyclic group of prime order q. Then, for picking up random g', h' from $\mathbb{G}'$, we can set $\mathsf{CHEval}(pk', m; r') = H'((g')^{H(m)}(h')^{r'})$, where $pk' = (g', h')$ and $sk' = \log_{g'}(h')$.

We construct a Tag-KEM scheme as follows:

- $\mathsf{Gen}(1^n)$: $\iota \leftarrow \mathcal{I} \cap \{0,1\}^n$; $H := H_\iota$; $x, y \leftarrow_R \mathbb{Z}/q\mathbb{Z}$; $X := g^x$; $Y := g^y$; $(pk', sk') \leftarrow \mathsf{CHGen}(1^n)$; return (pk, sk), where $pk = (e, g, g, X, Y, H, pk')$ and $sk = (pk, x, y)$.
- $\mathsf{Enc}(pk, \mathsf{tag}; r)$: $r \leftarrow \mathbb{Z}/q\mathbb{Z}$; $u := g^r$; $r' \leftarrow \mathtt{COIN}$; $t := \mathsf{CHEval}(pk', H(\mathsf{tag}, u); r')$; $\tau = (X^t Y)^r$; $C := (u, r', \tau)$; $K := X^r$; return (C, K).
- $\mathsf{Dec}(sk, \mathsf{tag}, C)$. Parse $C = (u, r', \tau)$; $t := \mathsf{CHEval}(pk', H(\mathsf{tag}, u); r')$; abort if $e(u, X^t Y) \neq e(g, \tau)$; otherwise, return $K := u^x$.

This Tag-KEM scheme is OW-ftCCA secure under the CDH assumption and publicly verifiable.

Σ-Protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$. Let $C = (u, r', \tau)$ such that $e(u, X^t Y) = e(g, \tau)$ for $t := \mathsf{CHEval}(pk', H(\mathsf{tag}, u); r')$. To prove $(C, K) \in \mathbb{R}^{\mathsf{priv}}_{pk,tag}$, where $C = (u, r', \tau)$, the prover picks up random $s, w \leftarrow (\mathbb{Z}/q\mathbb{Z})^\times$, computes $\hat{K} := K^s$ and $a = e(g, \hat{K})^w$, and sends $(\hat{K}, a)$ to the verifier. The verifier picks up random $c \leftarrow \mathbb{Z}/q\mathbb{Z}$ and sends c back. The prover answers $z := w + cs^{-1} \bmod q$. The verifier accepts if $e(g, \hat{K})^z = a \cdot e(u, X)^c$.

Note that $\mathsf{sim}\Sigma(C, c)$ can be computed as follows: The committer picks up random $\hat{K} \leftarrow G^\times = (G - \{1_G\})$; picks up random $z \leftarrow \mathbb{Z}/q\mathbb{Z}$; and outputs $a = e(g, \hat{K})^z \cdot e(u, X)^{-c}$. To open the commitment to c, he sends z to the verifier.

7.2 5-Round, Factoring-Based Implementation

We present a OW-ftCCA Tag-KEM based on the factoring assumption. This can be obtained by converting a OW-stCCA Tag-KEM [18,28] using a chameleon hash function. We also provide a Σ-protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$ and a $\hat{\Sigma}$-Protocol for $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$.

Let $n = PQ$ be a Blum integer for safe primes P, Q, i.e., $P, Q \equiv 3 \pmod 4$ where $p = (P-1)/2$ and $q = (Q-1)/2$ are both primes. Let $QR_n^+ \triangleq \{x \in \mathbb{Z}_n^* \mid (\frac{x}{P}) = (\frac{x}{Q}) = 1\}$. We assume that the factoring of N is hard. Let $\mathcal{H}$ be a keyed CR hash function family. Let $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ be a chameleon hash function. Its concrete implementation is, for instance, as follows: Let $H : \{0,1\}^* \to \{0,1\}^k$ and $H' : \mathbb{Z}/n\mathbb{Z} \to \{0,1\}^k$ be CR hash functions. Let $n' = p'q'$ be a composite number of large primes, p' and q'. Let $g' = g_0^{2^k}$ for some $g_0 \in (\mathbb{Z}/n'\mathbb{Z})^\times$, $pk' = (H, g', k)$, and $sk' = (p', q')$. Then define $\mathsf{CHEval}(pk', m; r') = H'((g')^{H(m)} r'^{2^k} \bmod n')$, where $r' \in (\mathbb{Z}/n'\mathbb{Z})^\times$.

We construct a Tag-KEM scheme as follows:

- $\mathsf{Gen}(1^\lambda)$: $\iota \leftarrow \mathcal{I} \cap \{0,1\}^\lambda$; $H := H_\iota$; $(n, (P, Q)) \leftarrow \mathsf{Gen}_{\mathsf{Blum}}(1^\lambda)$; pick up random $g \in QR_n^+$ such that $\#\langle g \rangle = pq$; $s \leftarrow [(n-1)/4]$; $Y := g^{2^{k+l} s}$; $(pk', sk') \leftarrow \mathsf{KGen}(1^\lambda)$; return (pk, sk), where $pk = (k, l, g, Y, H, pk')$ and $sk = (pk, s, P, Q)$.
- $\mathsf{Enc}(pk, \mathsf{tag})$: $r \leftarrow [(n-1)/4]$; $u := g^{2^{k+l} r}$; $r' \leftarrow \mathtt{COIN}$; $t := \mathsf{CHEval}(pk', H(\mathsf{tag}, u); r')$; $\tau := (g^t Y)^r$; $C := (u, r', \tau)$; $K := g^{2^l r}$; return (C, K).
- $\mathsf{Dec}(sk, \mathsf{tag}, C) = K$: Parse $C = (u, r', \tau)$; $t := \mathsf{CHEval}(pk', H(\mathsf{tag}, u); r')$; abort if $\tau^{2^k} \neq u^{2^{k+l} s + t}$; otherwise, compute $K := \sqrt[2^k]{u}$ (which is unique and easy to compute by using (P, Q) because of the property of Blum integers); return K.

This Tag-KEM scheme is OW-ftCCA under the factoring assumption [18,28].

Σ-protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$. Observe that $u = K^{2^k}$. There is a classic efficient Σ-protocol for $\mathbb{R}^{\mathsf{priv}}_{pk,tag}$, called the Guillou-Quisquater scheme [17]. Let $C = (u, \tau)$ such that $\tau^{2^k} = u^{2^{k+l} s + t}$. To prove $(C, K) \in \mathbb{R}^{\mathsf{priv}}_{pk,tag}$, where $C = (u, \tau)$, the prover picks up random $w \leftarrow \langle g \rangle$, computes $a = w^{2^k} \bmod N$, and sends a to

the verifier. The verifier picks up random $c \leftarrow [2^k]$ and sends c back. The prover answers $z := wK^c \bmod N$. The verifier accepts if $z^{2^k} = a \cdot u^c \pmod N$.

$\mathsf{sim}\Sigma(C, c)$ can be computed as follows: The committer picks up random $z \leftarrow \langle g \rangle$; and outputs $a = z^{2^k} u^{-c}$. To open the commitment to c, he sends z to the verifier.

$\hat{\Sigma}$-Protocol for $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$. Since this Tag-KEM scheme is not publicly verifiable, we require a weak extractable $\hat{\Sigma}$-protocol for $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$. What should be shown here is that $u = (g^{2^{k+l}})^r$ and $\tau = (g^t Y)^r$ have the same exponent r. Here we slightly relax $\mathbb{R}^{\mathsf{pub}}_{pk,tag}$ such that $\log_{g^{2^{k+l}}}(u) \equiv \log_{g^t Y}(\tau) \pmod{pq}$, which suffices for our purpose. Then we have the following $\hat{\Sigma}$-protocol.

- The prover chooses $w \in [N \cdot 2^{2l}]$ at random and sends $a_u = (g^{2^{k+l}})^w$ and $a_\tau = (g^t Y)^w$ to the verifier.
- The verifier chooses $c \in [2^l]$ at random and sends it to the prover.
- The prover sends back $z = w + c \cdot r \in \mathbb{Z}$.
- The verifier accepts if $(g^{2^{k+l}})^z = a_u u^c \pmod N$ and $(g^t Y)^z = a_\tau \tau^c \pmod N$.

We show that this protocol is weak extractable. Assume $r_u = \log_{g^{2^{k+l}}}(u) \neq \log_{g^t Y}(\tau) = r_\tau \pmod{pq}$. Let $w_u = \log_{g^{2^{k+l}}}(a_u) \bmod pq$ and $w_\tau = \log_{g^t Y}(a_\tau) \bmod pq$. Then for given c, one can verify that c is a value committed to in (a_u, a_τ), by checking $(\frac{u'}{\tau'})^e = \frac{a'_\tau}{a'_u} \pmod N$, where $u' = g^{r_u}$, $\tau' = g^{r_\tau}$, $a'_u = g^{w_u}$ and $a'_\tau = g^{w_\tau}$, which can be computed from primes, P, Q where $N = PQ$ is a William integer.

7.3 Comparison

The non-interactive SSTC schemes and their variants [14,8,15,21,16,10,22] are all constructed in the same framework: Let $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a digital signature

Table 1. Comparison with previous schemes

	Starting Scheme	Assumption	Reduction	Size [a]	Resulting Scheme
[14,21]	sig (DSA)	strong (DSA)	tight	short	efficient
[14,21,15,16]	sig ([6])	mild (sRSA)	loose	short	practical [b]
[14,21]	sig ([25])	weak (OWP[c])	loose	long	very inefficient
[15]	sig ([4])	mild (sDH)	tight	short	efficient
[10]	sig ([27])	weak (CDH)	very loose	long	efficient
[22]	sig ([19])	weak (RSA)	very loose	short	inefficient
This work (2-rnd)	KEM ([20,28])	weak (CDH)	tight	short	efficient
This work (5-rnd)	KEM ([18,28])	weak (Factoring)	loose	long [d]	less practical

[a] Total size of public parameters and commitments.
[b] Resulting scheme is not so efficient because committer should hash tag into set of primes.
[c] One-way permutation (OWP) is needed to construct Σ-protocol.
[d] IND-CCA2 PKEs based on factoring [18,28] need long public keys.

scheme that is existentially unforgeable against chosen message attacks (EUF-CMA). In addition, it must not be a one-time digital signature scheme. Let vk be a verification key of the signature scheme, which is seen as the CRS and let tag be a tag. To commit to m, a committer uses the simulator of a Σ-protocol on instance (vk, tag) and challenge m, for relation $R = \{((vk, tag), \sigma) \,|\, \exists r : \sigma = \mathsf{Sig}_{sk}(tag; r)\}$. Indeed the commitment is $a \leftarrow \mathsf{sim}\Sigma((vk, tag), m)$ for R. Hence, the only essential difference among the previous works depends on which signature scheme they used.

In the state-of-the-art techniques, it is more expensive to construct practical (non-one-time) EUF-CMA secure digital signatures than practical CCA secure public-key encryption schemes. We compare our instantiations with the previous results in Table 1.

References

1. Abe, M., Gennaro, R., Kurosawa, K.: Tag-KEM/DEM: A new framework for hybrid encryption. Journal of Cryptology 21(1), 97–130 (2008)
2. Anada, H., Arita, S.: Identification Schemes from Key Encapsulation Mechanisms. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 59–76. Springer, Heidelberg (2011)
3. Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.: Identification Protocols Secure against Reset Attacks. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 495–511. Springer, Heidelberg (2001)
4. Boneh, D., Boyen, X.: Short Signatures without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
5. 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)
6. Cramer, R., Shoup, V.: Signature schemes based on the strong rsa assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)
7. Damgård, I., Jurik, M.: A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 125–140. Springer, Heidelberg (2001)
8. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: STOC 2003, pp. 426–437 (2003)
9. Damgård, I., Nielsen, J.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
10. Dodis, Y., Shoup, V., Walfish, S.: Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 515–535. Springer, Heidelberg (2008), http://www.shoup.net/papers/gucc.pdf
11. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM. J. Computing 30(2), 391–437 (2000); (Presented in STOC 1991)
12. Feige, U., Shamir, A.: Zero-Knowledge Proofs of Knowledge in Two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)

13. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, STOC 1990 (1990)
14. Garay, J.A., Mackenzie, P., Yang, K.: Strengthening Zero-Knowledge Protocols using Signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 177–194. Springer, Heidelberg (2003)
15. Gennaro, R.: Multi-Trapdoor Commitments and their Applications to Proofs of Knowledge Secure under Concurrent Man-in-the-Middle Attacks. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 220–236. Springer, Heidelberg (2004), http://eprint.iacr.org/2003/214
16. Groth, J.: Honest Verifier Zero-Knowledge Arguments Applied. PhD thesis, Basic Resarch in Computer Science, University of Aarhus (2004)
17. Guillou, L.C., Quisquater, J.-J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
18. Hofheinz, D., Kiltz, E.: Practical Chosen Ciphertext Secure Encryption from Factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
19. Hohenberger, S., Waters, B.: Short and Stateless Signatures from the RSA Assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
20. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
21. MacKenzie, P.D., Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004)
22. Nishimaki, R., Fujisaki, E., Tanaka, K.: A Multi-Trapdoor Commitment Scheme from the RSA Assumption. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 182–199. Springer, Heidelberg (2010)
23. Ostrovsky, R., Persiano, G., Visconti, I.: Simulation-Based Concurrent Non-Malleable Commitments and Decommitments. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 91–108. Springer, Heidelberg (2009)
24. 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)
25. Rompel, J.: One-way functions are necessary and sufficient for secure signature. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 1990), pp. 387–394 (1990)
26. Shoup, V.: A proposal for an ISO standard for public key encryption. Technical report, Cryptology ePrint Archive, Report 2001/112 (December 2001)
27. Waters, B.: Efficient Identity-Based Encryption without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
28. Wee, H.: Efficient Chosen-Ciphertext Security Via Extractable Hash Proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010)

A Definitions

A.1 Chameleon Hash Function

A chameleon hash function $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ consists of three algorithms: CHGen is a PPT algorithm that takes as input security parameter 1^κ and outputs a pair of public and trap-door keys (pk, tk). CHEval is a PPT algorithm that takes as input pk and message $m \in \{0,1\}^*$, drawing random r from coin space $\mathtt{COIN}_{pk}$, and outputs chameleon hash value $c = \mathsf{CHEval}(pk, m; r)$. Here $\mathtt{COIN}_{pk}$ is uniquely determined by pk. CHColl is a DPT algorithm that takes as input (pk, tk), $m, m' \in \{0,1\}^*$ and $r \in \mathtt{COIN}_{pk}$, and outputs $r' \in \mathtt{COIN}_{pk}$ such that $\mathsf{CHEval}(pk, m; r) = \mathsf{CHEval}(pk, m'; r')$. We require that for every (pk, tk) generated by $\mathsf{CHGen}(1^\kappa)$, every $m, m' \in \{0,1\}^*$, and every $r \in \mathtt{COIN}_{pk}$, there exists a *unique* $r' \in \mathtt{COIN}_{pk}$ such that $\mathsf{CHEval}(pk, m; r) = \mathsf{CHEval}(pk, m'; r')$, and $\mathsf{CHColl}(pk, tk, m, m', r)$ always computes r' in time $poly(\kappa + |m| + |m'|)$. We say that $\mathcal{CH}$ is collision-resistance if for every non-uniform PPT adversary A,

$$\Pr\left[\begin{array}{l}(pk, tk) \leftarrow \mathsf{CHGen}(1^\kappa);\ (m_1, m_2, r_1, r_2) \leftarrow A(pk) : \\ \mathsf{CHEval}(pk, m_1; r_1) = \mathsf{CHEval}(pk, m_2; r_2) \ \wedge\ (m_1 \neq m_2)\end{array}\right] = \mathsf{negl}(\kappa).$$

A.2 Man-In-The-Middle Attacks

Let (A, B) be a two-party (interactive) protocol between A and B. The *man-in-the-middle execution* is defined on any two-party protocol. In the man-in-the-middle execution, the man-in-the-middle adversary $\mathcal{A}$ is simultaneously participating in (unbounded polynomially many) multiple concurrent executions of protocol (A, B). Executions in which $\mathcal{A}$ is playing the role of B are said to belong to the *left* interaction, whereas the executions in which $\mathcal{A}$ is playing the role of A are said to belong to the *right* interaction. In the cMiM execution, adversary $\mathcal{A}$ may start executions in both left and right interactions and may interact with both left and right parties in arbitrarily interleaved order of messages. We say that a cMiM execution is *left-concurrent* if $\mathcal{A}$ may interact with multiple copies of A on (possibly different) common inputs x_i in the left interaction, whereas $\mathcal{A}$ may interact with only one copy of B on common input x' in the right interaction. We say that a cMiM execution is *fully-concurrent* if $\mathcal{A}$ may interact with multiple copies of A on (possibly different) common inputs x_i in the left interaction and $\mathcal{A}$ may interact with with multiple copies of B on (possibly different) common inputs x'_i in the right interaction.

A.3 Tag-Based Identification Schemes

Tag-Based ID Schemes. A tag-based identification scheme $\mathsf{ID} = (\mathsf{KGen}, P, V)$ consists of a PPT algorithm KGen and a pair of probabilistic polynomial-time interactive algorithms, (P, V) , where for every (sufficiently large) $n \in \mathbb{N}$, KGen takes 1^n and outputs (pk, sk), and (P, V) is an interactive protocol between P and V, where, $(p(sk), V)(pk, \mathsf{tag})$ denotes the random variable of one bit

outputted by V, representing whether V accepts or rejects the communication after P took (pk, tag, sk) and V took (pk, tag) and they run one execution of the protocol. For completeness, we require that for every $(pk, sk) \in \mathsf{KGen}(1^n)$ and every $\mathsf{tag} \in \{0,1\}^n$, it always holds that $(P(sk), V)\ (pk, \mathsf{tag}) = 1$ for honest P and V.

cMiM Security of Tag-Based ID Schemes. We define the concurrent man-in-the-middle security (cMiM) of a tag-based identification scheme, as follows. Let A be a cMiM adversary for a tag-based ID scheme. The action of A in this case is the same as that in the non-tag case, except that A can activate P and V with *his chosen tags*. The advantage of A is defined as the probability that $(A^{P(sk)}, V)(pk, \mathsf{tag}) = 1$ for tag that A has chosen in the right interaction, and any tag in the left interaction should be different from the tag tag. A tag-based ID scheme is cMiM secure if for every adversely polynomial-size circuit ensemble A, $\mathsf{Adv}^{\mathsf{cMiM}}_{A,\mathsf{ID}}(n) = \mathsf{negl}(n)$, where $\mathsf{Adv}^{\mathsf{cMiM}}_{A,\mathsf{ID}}(n) \triangleq \Pr[(pk, sk) \leftarrow \mathsf{KGen}(1^n) : (A^{P_{sk}}, V)(pk, \mathsf{tag}) = 1]$.

A First-Order Leak-Free Masking Countermeasure

Houssem Maghrebi[1], Emmanuel Prouff[2], Sylvain Guilley[1,3], and Jean-Luc Danger[1,3]

[1] TELECOM-ParisTech, Crypto Group,
37/39 rue Dareau, 75 634 PARIS Cedex 13, France
{maghrebi,guilley,danger}@telecom-paristech.fr
[2] Oberthur Technologies,
71-73, rue des Hautes Pâtures 92726 Nanterre Cedex, France
e.prouff@oberthur.com
[3] Secure-IC S.A.S.,
2 rue de la Châtaigneraie, 35 576 CESSON SEVIGNÉ, France

Abstract. One protection of cryptographic implementations against side-channel attacks is the masking of the sensitive variables. In this article, we present a first-order masking that does not leak information when the registers change values according to some specific (and realistic) rules. This countermeasure applies to all devices that leak a function of the distance between consecutive values of internal variables. In particular, we illustrate its practicality on both hardware and software implementations.

Moreover, we introduce a framework to evaluate the soundness of the new first-order masking when the leakage slightly deviates from the rules involved to design the countermeasure. It reveals that the countermeasure remains more efficient than the state-of-the-art first-order masking if the deviation from the ideal model is equal to a few tens of percents, and that it is as good as a first-order Boolean masking even if the deviation is 50%.

Keywords: First-order masking, leakage in distance, leakage-free countermeasure.

1 Introduction

During the last ten years, a lot of efforts have been dedicated towards the research about side-channel attacks [9, 1] and the development of corresponding countermeasures. In particular, there have been many endeavors to develop effective countermeasures against differential power analysis (DPA) [10] attacks. DPA, and more generally side channel analysis (SCA for short), take advantage of the fact that the power consumption of a cryptographic device depends on the internally used secret key. Since this property can be exploited with relatively cheap equipments, DPA attacks pose a serious practical threat to cryptographic devices, like smart cards (ASICs) or embedded systems (DSPs, CPUs and FPGAs).

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 156–170, 2012.

A very common countermeasure to protect implementations of block ciphers against SCA is to randomize the sensitive variables by masking techniques. The idea of masking the intermediate values inside a cryptographic algorithm has been suggested in several papers [8,5] as a possible countermeasure to power analysis attacks. The technique is generally applicable if all the fundamental operations used in a given algorithm can be rewritten in the masked domain. This is easily seen to be the case in classical algorithms such as the DES or AES. Masking ensures that the sensitive data manipulated by the device is masked with at least one random value so that the knowledge on a subpart of the shares (the masked data or the mask) does not give information on the sensitive data itself.

The masking can be characterized by the number of random masks used per sensitive variable. So, it is possible to give a formal definition for a high-order masking scheme: a d^{th}-order masking scheme involves $d+1$ shares. The security is reached at order d provided that any combination of d intermediate variables during the entire computation conveys no information about the sensitive variable.

We must concur that computing with $d+1$ shares without revealing information from any set of size d of intermediate values can be challenging. Some first-order masking techniques have been successfully proved to be secure against first-order SCA (1O-SCA) attacks. Nonetheless, masked implementations can always be attacked, since all shares [7] or a judicious combination [16] of them unambiguously leaks information about the sensitive variable. The construction of an efficient d^{th}-order masking scheme thus became of great interest. One sound solution has been put forward recently in [17].

In this paper, we are not concerned with higher-order masking, but devise optimised masking schemes when the leakage function is known. Typically, we show that with $d = 1$, and assuming a Hamming distance leakage function (or even some small variations of it), it is possible to zero the sensible information leaked during the registers update.

The rest of the paper is structured as follows. In Sec. 2, the concept of Boolean masking and 1O-SCA are formally defined. We also introduce some useful notations. The most critical part when securing implementations is to protect their non-linear operations (*i.e.* the sbox calls). In Sec. 3, we recall the methods which have been proposed in the literature. Then, we introduce a new and a simple countermeasure which counteracts 1O-SCA when the device leaks the Hamming distance. The security analysis is conducted for both idealized model in Sec. 4 and imperfect model in Sec. 5. The conclusion and some perspectives are in Sec. 6. Simulation results in the imperfect model are in appendix A.

2 Preliminaries

In this paper we focus on the Boolean masking countermeasure [5,8]. Its idea is to mask the sensitive data (which depends on both plaintext and the secret key) by a *XOR* operation (denoted $\oplus$) with a random word, in order to avoid

the correlation between the cryptographic device's power consumption and the data being processed. The main difficulty of masking resides in the handling of shares of a unique intermediate variable through a non-linear function (*i.e.* the cipher sboxes). An $n \times m$ sbox (*i.e.* with n input bits and m output bits) can be described as a vectorial output mapping $F : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, or a collection of Boolean functions $F = (f_1, \cdots, f_m)$, where each f_i is defined from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. The vectorial function F is also called an (n, m)-function. A (n, m)-function F such that every element $Y \in \mathbb{F}_2^m$ has exactly 2^{n-m} pre-images by F is said to be *balanced*. Its outputs are uniformly distributed over $\mathbb{F}_2^m$ when its inputs are uniformly distributed over $\mathbb{F}_2^n$. As recalled in introduction, the manipulation of sensitive data through this function may be protected using d^{th}-order masking scheme. When such a scheme is applied, it is expected that no HO-SCA of order less than or equal to d is successful. The order relates to the number of different instantaneous leakages considered by the attack. In this paper, we focus on first-order masking scheme secure against 1O-SCA. For the rest of the paper, we adopt the following notations. Random variables are printed in capital letters (*e.g.* X), whereas their realization is noted with small letters (*e.g.* x), and their support by calligraphic letters (*e.g.* $\mathcal{X}$). The mutual information between X and Y is denoted $\mathsf{I}[X; Y]$; it measures the mutual dependency of the two variables. The Hamming weight of x, written as $\mathsf{HW}(x)$, is the number of ones in the binary word x.

3 Secure Computation against 1O-DPA Using ROMs

Let X and K denote two random variables respectively associated with some plaintext subpart values x and a secret sub-key k manipulated by a cryptographic algorithm. Let us moreover denote by Z the sensitive variable $X \oplus K$. When a *first-order* Boolean masking is involved to secure the manipulation of Z, the latter variable is randomly split into two shares M_0, M_1 such that:

$$Z = M_0 \oplus M_1 \ . \tag{1}$$

The share M_1 is usually called *the mask* and is a random variable uniformly distributed over $\mathbb{F}_2^n$. The share M_0, called *the masked variable*, plays a particular role and is built such that $M_0 = Z \oplus M_1$. Variables Z and M_1 are assumed to be mutually independent. To enable the application of a transformation S on a variable Z split in two shares, as in (1), a so-called first-order masking scheme must be designed. It leads to the processing of two new shares M_0' and M_1' such that:

$$S(Z) = M_0' \oplus M_1' \ . \tag{2}$$

Once again, the share M_1' is usually generated at random and the share M_0' is defined such that $M_0' = S(Z) \oplus M_1'$. The critical point is to deduce M_0' from M_0, M_1 and M_1' without compromising the security of the scheme (*w.r.t.* 1O-SCA). When S is linear for the law $\oplus$, then deducing M_0' is an easy task. Actually, since the relation $S(Z) = S(M_0 \oplus M_1) = S(M_0) \oplus S(M_1)$ holds, then M_0' can be simply chosen such that $M_0' = S(M_0) \oplus S(M_1) \oplus M_1'$.

When S is non-linear for the law $\oplus$ (which occurs when S is a sbox), achieving first-order security is much more difficult. The latter security indeed implies that no instantaneous leakage during the processing leaks information on Z and hence, particular attention must be paid on each elementary calculus or memory manipulation. Several solutions have been proposed to deal with this issue. Commonly, there are three strategies [15]:

1. *The re-computation method* [2,11]: this technique involves the computation of a precomputed table corresponding to the masked sbox and the generation of one or several random value(s). In its most elementary version, two random values M_1 and M'_1 are generated and the table $T^\star$ representing the function $S' : Y \mapsto S(Y \oplus M_1) \oplus M'_1$ is computed from S and stored in RAM. Then, each time the masked variable M'_0 has to be computed from the masked input $Z \oplus M_1$, the table $T^\star$ is accessed.
2. *Global Look-up Table* [15,24]: this method also involves the computation of a precomputed look-up table, denoted $T^\star$, associated to the function $(X, Y, Y') \mapsto S(X \oplus Y) \oplus Y'$. To compute the masked variable M'_0, the global look-up table method (GLUT for short) performs a single operation: the table look-up $T^\star[Z \oplus M_1, M_1, M'_1]$. The main and important difference with the first method is that the value $S(X \oplus Y) \oplus Y'$ has been precomputed for every possible 3-tuple of values. Consequently, there is no need to re-compute before each algorithm processing and it can be stored in ROM[1]. In a simplified version (sufficient to thwart only 1O-SCA), the output mask and the input mask are chosen equal. In this case, the dimension of the table is $2n$ instead of $3n$ and the table look-up becomes $T^\star[Z \oplus M_1, M_1]$. We consider this latter version of the GLUT method in the following.
3. *The sbox secure calculation* [17,5,14,18,26]: the sbox outputs are computed *on-the-fly* by using a mathematical (*e.g.* polynomial) representation of the sbox. Then, each time the masked value M'_0 has to be computed, an algorithm performing S and parametrized by the 3-tuple (M_0, M_1, M'_1) is executed. The computation of S is split into elementary operations (bitwise addition, bitwise multiplication, ...) performed by accessing one or several look-up table(s).

Moreover, depending on the number of masks generated to protect the sbox calculations, we can distinguish two modes of protections:

1. *The single mask protection mode*: in this mode, every computation $S(Z)$ performed during the execution is protected with a single pair of input/output masks (M_1, M'_1).
2. *The multi-mask protection mode*: in this mode, the pair of masks (M_1, M'_1) is re-generated each time a computation $S(Z)$ must be protected and thus many times per algorithm execution.

In [15], the authors have shown that the choice between the three methods depends on the protection mode in which the algorithm is implemented. In fact,

[1] Recall that in embedded systems, ROM is a much less costly resource than RAM.

when the algorithm is protected in the single-mask protection mode, the re-computation method is more appropriate and induces a smaller timing/memory overhead. In the multi-mask protection mode, the re-computation method is often much more costly since the recomputation must be done before every sbox processing. Moreover, in both contexts it requires 2^n bytes of RAM to be free, which can be impossible in some very constrained environments. Concerning the sbox secure computation, it is secure against first-order SCA and does not need particular RAM allocation. However, it is often more time consuming than the first two methods and can only be used to secure sboxes with a simple algebraic structure (as *e.g.* the AES or the SEED sboxes). Regarding the GLUT method, it seems at a first glance to be the most appropriate method. Its timing performances are ideal since it requires only one memory transfer. Moreover, it can be applied in both protection modes described above. From a security point of view, the GLUT method has however a flaw since it manipulates the masked data $Z \oplus M_1$ and the mask M_1 at the same time. Actually, $Z \oplus M_1$ and M_1 are concatenated to address the look-up table $T^\star$ and thus, the value $Z \oplus M_1 \,||\, M_1$ is transferred through the bus. Since the latter variable is statistically dependent on Z, any leakage on it is potentially exploitable by a first-order DPA involving the higher-order moments of the concatenated random variable. It must however be noted that such a leakage on the address does not necessarily occurs during the bus transfers or the registers updates. Indeed, when for instance the latter ones leak the Hamming weight between an independent and random initial state and the address $Z \oplus M_1||M_1$, then the leakage is independent on Z and no first-order DPA is hence applicable. This example shows the importance of the device architecture when assessing on a countermeasure soundness. In this paper, we focus on the GLUT method. Our proposal is to benefit from all the seminal assets of the method and to additionally achieve first-order security for some realistic architectures (including the Von-Neumann ones).

3.1 Detailed Description of GLUT Method

In hardware, GLUT method can be implemented as shown in Fig. 1. This figure encompasses the masking scheme already presented in [25]. For the sake of simplicity, the linear parts, like the expansion (in DES), MixColumns (in AES), *etc.* are not represented. So, without loss of generality, we assume that the sbox S in an (n, n)-function. For instance, using AES, n can be chosen equal to 8 (straightforward tabulation of SubBytes), 4 (with the decomposition of SubBytes in $\mathrm{GF}((2^4)^2)$), or even 2 (using the $\mathrm{GF}(((2^2)^2)^2)$ tower field [19]). The registers R and M contain respectively the masked variable and the mask.

For any (n, n)-function S that must be processed in a secure way, the core principle is to define from S the lookup table representation of a new $(3n, n)$-function S' which is indexed by both the masked data and the masking material. Thanks to this new function, a masked representation $S(Z) \oplus M'_1$ of $S(Z)$ is securely derived from $Z \oplus M_1$, M_1 and the output mask M'_1 by accessing the look-up table representing S'. The size of the table can be reduced by defining the output mask as a deterministic function of the input mask. In such a case, the

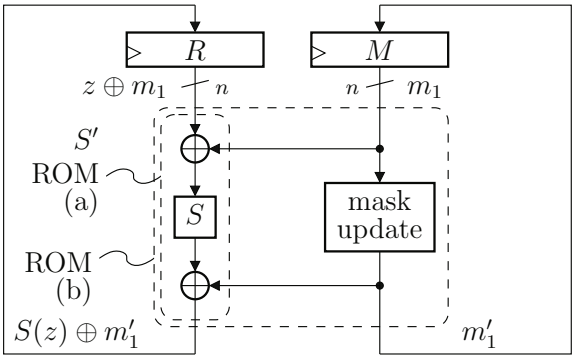


Fig. 1. First-order hardware masking implementation

ROM lookup-table represents a $(2n, n)$-function S' such that $S'(Z \oplus M, M) = S(Z) \oplus M'$, where M' is a deterministic function of M (*e.g.* $M' = M \oplus \alpha$ for some constant α).

In the first case, the ROM look-up table has $(3n)$-bit input words: the two shares and the new mask for the remasking, and one n-bit output (*e.g.* option (a) of Fig. 1). In the second case, the new masks are derived deterministically from the old ones, and thus the ROM look-up table can have only the two input shares as inputs (*e.g.* option (b) of Fig. 1). The ROM look-up table thus represents a $(2n, 2n)$-function. This is the scenario we consider in the rest of this article.

3.2 Leakage of the ROM-Based 1O-DPA Protection Implementation

During the processing of the scheme depicted in Fig. 1, we assume that only the updating of the registers R and M leak information. Indeed, since the leakage at the register level is perfectly synchronized with the system clock, it has a relatively high density of energy which is easily detectable. On the other hand, the leakage from the combinational logic is very dependent on the implementation and spreads over the time. It can be seriously reduced by taking advantage of the ROM tables [22]. In the following, we denote by L_R and L_M the leakage variables corresponding to the updating of the registers R and M respectively. We have:

$$\begin{aligned} L_R &= A(Z \oplus M_1, Z' \oplus M'_1) + N_R \\ L_M &= A(M_1, M'_1) + N_M \ , \end{aligned} \tag{3}$$

where A is a deterministic function representing the power consumption during the register updating and where N_R and N_L are two independent noises. The power consumption related to the simultaneous updating of the registers R and M equals $L_R + L_M$ and is denoted by O. In a first time, we assume that A in (3) has the following property that will be relaxed in the second part of this paper.

Property 1. For any pair (X, Y)*, we have* $A(X, Y) = A(X \oplus Y)$.

Remark 1. Many security analyses in the literature have been conducted in the so-called Hamming Distance model [12,3]. In this model, the function A is assumed to be the Hamming distance between X and Y and thus clearly satisfies Property 1.

When A satisfies Property 1, the variable O satisfies:

$$O = A(\Delta(Z) \oplus \Delta(M)) + A(\Delta(M)) + N_R + N_M \ , \tag{4}$$

where $\Delta(Z)$ and $\Delta(M)$ respectively denote $Z \oplus Z'$ and $M_1 \oplus M_1'$.

The distribution of O (and in particular its variance) depends on the sensitive variable $\Delta(Z)$. This dependency has already been exploited in several attacks (*e.g.* [28]). In this paper, we study whether it can be broken by replacing the bitwise data masking $Z \oplus M_1$ by a new one denoted by $Z@M_1$ and by adding conditions on M_1 and M_1'.

3.3 Towards a New Masking Function

A simple solution, deeply analyzed in this paper, is to choose a function @ such that $Z \ @ \ M_1 = Z \ \oplus \ F(M_1)$ for some well chosen function F. For such a new masking function, @ is not commutative and M_1 and Z do no longer need to have the same dimension n. Only the output size of the function F must be n. In the following, we denote by p the dimension of M_1 and we assume that F is a (p, n)-function. We will see in Sec. 4.1 that p and n must satisfy some conditions for the masking to be sound. In this case, the deterministic part in (4) can be rewritten:

$$\begin{aligned} & A(Z@M_1, Z'@M_1') + A(M_1, M_1') \\ \doteq\ & A(Z \oplus Z' \oplus F(M_1) \oplus F(M_1')) + A(M_1 \oplus M_1') \\ =\ & A(\Delta(Z) \oplus F(M_1) \oplus F(M_1')) + A(\Delta(M_1))\,. \end{aligned} \tag{5}$$

In view of (5), we deduce the two following sufficient conditions for O to be independent of $\Delta(Z)$:

1. [**Constant Masks Difference**]: $M_1 \oplus M_1'$ is constant and
2. [**Difference Uniformity**]: $F(M_1) \oplus F(M_1')$ is uniform.

To the two security conditions above, a third one must also be introduced to enable the bitwise introduction of the key on the internal state X:

3. [**Operations Commutativity**]: For every (X, M_1, K), we have:

$$X@M_1 \oplus K = (X \oplus K)@M_1 \ .$$

In the following section, we propose a way to specify M_1, M'_1 and F to satisfy the three sufficient conditions. We structure our study of this new technique in two steps: the first one (*cf.* Sec. 4) is performed by assuming that A satisfies Property 1 (*i.e.* $A(X,Y) = A(X \oplus Y)$) and the second one (*cf.* Sec. 5) is conducted in an imperfect model where A satisfies $A(X,Y) = P(X,Y)$, with $P(X,Y)$ being a polynomial function in $\mathbb{R}[X_1, \cdots, X_n, Y_1, \cdots, Y_n]$ where X_i and Y_i denote the i^{th} Boolean coordinate of X and Y respectively.

4 Study in the Idealized Model

4.1 Our Proposal

Under Property 1 and as argued in the previous section, we can render the variable O independent of $\Delta(Z)$. It indeed amounts to fix the condition $M'_1 = M_1 \oplus \alpha$ for some nonzero constant term α and to design a function F *s.t.* the function $Y \mapsto F(Y) \oplus F(Y \oplus \alpha)$ is uniform for this α. The latter function is usually called derivative of F with respect to α. The construction of functions F having such uniform derivatives has been highly investigated in the literature [4, Chp. 4].

We give hereafter two examples of construction of such functions F.

First Construction Proposal: we choose $p = n + 1$ and we split $\mathbb{F}_2^{n+1}$ into the direct sum $E \oplus (E \oplus \alpha)$, where E is a n-dimensional vector space and $\alpha \in \mathbb{F}_2^p$. One bijective function G from E into $\mathbb{F}_2^n$ is arbitrarily chosen and F is defined such that for every $Y \in \mathbb{F}_2^{n+1}$, we have $F(Y) = G(Y)$ if $Y \in E$ and $F(Y) = 0$ otherwise.

Second Construction Proposal: we choose $p = n + n'$ with $n' < n$ and we select one injective function G from $\mathbb{F}_2^{n'}$ into $\mathbb{F}_2^n - \{0\}$. Then, for every $(X,Y) \in \mathbb{F}_{2^{n'}} \times \mathbb{F}_{2^n} = \mathbb{F}_{2^p}$ we define $F(X,Y) = G(X) \cdot Y$ with $\cdot$ the field product over $\mathbb{F}_{2^n}$. The outputs of the (p,n)-function F are uniformly distributed over $\mathbb{F}_2^n$ (since the functions $Y \mapsto G(X) \cdot Y$ are linear and non-zero for every X). Moreover, for every non-zero element α' in $\mathbb{F}_{2^{n'}}$, the function $D_\alpha F$ defined with respect to $\alpha = (\alpha', 0) \in \mathbb{F}_{2^{n'}} \times \mathbb{F}_{2^n}$ is also balanced. Indeed, we have $D_\alpha F = (G(X) \oplus G(X + \alpha')) \cdot Y$ and, since the injectivity of G implies that $G(X) \oplus G(X \oplus \alpha')$ is never zero, the functions $Y \mapsto (G(X) \oplus G(X \oplus \alpha')) \cdot Y$ are linear and non-constant for every X.

The two constructions of F satisfy the *difference uniformity* condition defined in Sec. 3.3. The mask dimension p for the first construction is only slightly greater than the dimension n of the data to be masked. This makes it more efficient than the second construction. However, the second construction ensures that not only $D_\alpha F$ but also F is balanced. This is not mandatory to ensure the security of the countermeasure in our context where the targeted leakage is assumed to satisfy Property 1, but it can be of interest if one wishes that the data Z and Z' be masked with a uniform mask $F(M_1)$ and $F(M'_1)$ respectively.

Figure 2 shows a hardware implementation of our countermeasure. The registers R and M contain respectively the masked variable $Z \oplus F(M_1)$ and the mask M_1. The mask update operation is constrained to be a $\oplus$ operation with

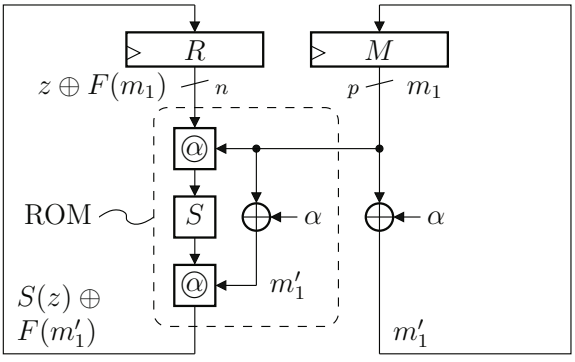


Fig. 2. Leak-free masking hardware implementation

a constant value α in order to satisfy the first condition. Consequently, every computation in the algorithm is protected with the single pair of masks (M_1, $M'_1 = M_1 \oplus \alpha$). Nonetheless, the value of M_1 changes at every computation; thus, the injected entropy in one computation is p bits. The table $T^\star$ representing the function $S' : (X, Y) \mapsto S(X@Y)@(Y \oplus \alpha) = S(X \oplus F(Y)) \oplus F(Y \oplus \alpha)$ has been pre-computed and stored in ROM. The new masked variable $S(Z) \oplus F(M'_1)$ is got by accessing the ROM table $T^\star$ as described in Fig. 2. We assume that this address is not leaking sensitive information but the leakage comes from the updating of the registers R and M following Equations (4) and (5).

4.2 Security Evaluation

In our security analysis, we assume that the attacker can query the targeted cryptographic primitive with an arbitrary number of plaintexts and obtain the corresponding physical observations, but cannot choose its queries in function of the previously obtained observations (such a model is called *non-adaptive known plaintext model* in [23]). We also assume that the attacker has access to the power consumption and electromagnetic emanations of the device and applies a first-order DPA attack but is not able to perform HO-DPA.

Regarding the leakage model, we assume that the device leaks a function A of the distance between the processed data and its initial state handled in the register (*i.e.* A satisfies Property 1). This situation is more general than the Hamming distance model, and notably encompasses the imperfect model studied in [27, Sec. 4]. The mutual information $\mathsf{I}[A(\Delta(Z) \oplus F(M_1) \oplus F(M'_1)) + A(\Delta(M)); \Delta(Z)] = 0$ since $\Delta(M)$ is constant and since $F(M_1) \oplus F(M'_1)$ is uniformly distributed over $\mathbb{F}_2^n$ and independent of $\Delta(Z)$. Hence, our construction is *leak-free* and immune against first-order attacks. Furthermore, as $A(\Delta(M))$ is constant, the mutual information

$$\mathsf{I}[A(\Delta(Z) \oplus F(M_1) \oplus F(M'_1)), A(\Delta(M)); \Delta(Z)]$$

is also null, which means that the masking countermeasure is secure against an adversary who observes the leakage in the transition from one state during

the registers update and can repeat this as many times as he wants. The adversary recovers the observations of the variable $(L_R + L_M)$ and can make all the treatments he wants (*e.g.* computation of mutual information univariate or multivariate, raise to any power the variable $L_R + L_M, \ldots$).

4.3 Application to the Software Implementation Case

Our proposal can be applied also in some particular software implementations. In some access memory schemes, the address and the value read are transferred through the same bus (*e.g.* Von-Neumann architecture). Thus, when accessing a table, the value overwrites the address and a leakage as in (4) occurs. Such access is obtained with a code such as:

```
mov  dptr, #tab
mov  acc, y
movc acc, @acc+dptr
```

In the code above, `dptr` refers to a data memory pointer and `#tab` to the address of a table stored in data. The variable `y` is assumed to contain the index of the value that must be read in table `tab`. After the third step, the accumulator register `acc` contains the value `tab[y]`. During this processing, the accumulator goes from state `y` to state `tab[y]`. Let us now assume that `tab` refers to the table T' defined in Sec. 4.1 and that `y` refers to the variable $(Z@M_1, M_1)$. If we associate the most significant bits of the accumulator `acc` to a (sub-)register R and its least significant bits to a (sub-)register M then we are in the same context as the analysis conducted in Secs. 4.1 and 4.2. A first-order DPA attack can be conducted on this register to reveal information about the sensitive data. Taking advantage from our proposal, the memory access is made completely secure under the assumption of Property 1.

5 Study in the Imperfect Model

It must first be remarked that the countermeasure proposed in the previous sections stays valid if $A(X, Y)$ can be rewritten under the form $P(X_1 \oplus Y_1, \cdots, X_n \oplus Y_n)$ with P being any polynomial defined over $\mathbb{F}_2^n$ with real coefficients.

In this section we assume that the hardware has been protected under the assumption that A satisfies Property 1, while the assumption is wrong. Namely, A was assumed to be s.t. $A(X, Y) = A(X \oplus Y)$ whereas in reality, it is a polynomial $P(X_1, \cdots, X_n, Y_1, \ldots, Y_n)$ that does not satisfy this property. In the following, we study experimentally the amount of information that the pair (L_R, L_M) defined in (3) leaks on (Z, Z') in this context where P is of (multivariate) degree d.

We recall that a polynomial of degree d in $\mathbb{R}[X_1, \cdots, X_n, Y_1, \cdots, Y_n]$ takes the following form:

$$P(X_1, \cdots, X_n, Y_1, \cdots, Y_n) = \sum_{\substack{(u,v) \in \mathbb{F}_2^n \times \mathbb{F}_2^n, \\ \mathsf{HW}(u) + \mathsf{HW}(v) \leq d}} a_{(u,v)} X_1^{u_1} \cdots X_n^{u_n} Y_1^{v_1} \cdots Y_n^{v_n} \ ,$$

where the $a_{(u,v)}$ are real coefficients. This leakage formulation is similar to that of the high-order stochastic model [21]. For example, it is shown in [16, Eqn. (3)] that $P(X_1, \cdots, X_n, Y_1, \cdots, Y_n)$ is equal to $\mathsf{HW}(X \oplus Y)$ when the coefficients $a_{(u,v)}$ satisfy:

$$a^{\mathrm{HD}}_{(u,v)} = \begin{cases} +1 & \text{if } \mathsf{HW}(u) = 1 \text{ and } v = 0, \\ +1 & \text{if } u = 0 \text{ and } \mathsf{HW}(v) = 1, \\ -2 & \text{if } \mathsf{HW}(u) = 1 \text{ and } v = u, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In the following experiment, we compute the mutual information between (L_R, L_M) and (Z, Z') when $d \leq 2$ or 3 and when the coefficients $a_{(u,v)}$ deviate randomly from those of (6)[2]. More precisely, the coefficients $a_{(u,v)}$ are drawn at random from this law:

$$\begin{array}{l} a_{(u,v)} \sim a^{\mathrm{HD}}_{(u,v)} + \mathcal{U}(\left[-\frac{\text{deviation}}{2}, +\frac{\text{deviation}}{2}\right]) \ , \\ a_{(u,v)} = 0 \quad \text{if} \quad \mathsf{HW}(u,v) > d \ . \end{array} \tag{7}$$

The randomness lays in the uniform law $\mathcal{U}(\left[-\frac{\text{deviation}}{2}, +\frac{\text{deviation}}{2}\right])$, that we parametrize by deviation $\in \{0.1, 0.2, 0.5, 1.0\}$. The low deviation values (such as 0.1 or 0.2) are realistic in hardware, as attested by [13]; in this paper, the leakage captured by a tiny coil has been shown to differ from the Hamming distance model by 17%. We thus consider that a a deviation of $\approx [10, 20]\%$ is representative of the hardware imperfections or on the model bias by integrated probes. A deviation of 1 has the same order of magnitude as the actual coefficients in (6); it indicates that the Hamming distance model is an incorrect hypothesis. Nonetheless, this case is very unlikely: indeed, the designer of the countermeasure can be expected to know (or to have checked) that the circuit leaks approximately in Hamming distance. Eventually, the deviation 0.5 represents an intermediate case: the leakage model is in-between an approximate Hamming distance model and a full random leakage model. The computed mutual information is $\mathsf{I}[O; Z, Z']$, where $O = P(Z \oplus F(M), Z' \oplus F(M \oplus \alpha)) + N_R + P(M, M \oplus \alpha) + N_M$. Therefore O is a RV, sum of a function of Z, Z', M and of $N_R + N_M \sim \mathcal{N}(\mu_R + \mu_M, \sigma_R^2 + \sigma_M^2) = \mathcal{N}(\mu, \sigma^2)$, a normal law. The simulation parameters and the results are shown in Appendix A.

It appears that the degree d has minor influence on the leakage. The major factor is the deviation from the Hamming distance model. As expected, for low deviations (much smaller than 1, *e.g.* 10% or 20%), the one-mask countermeasure (abridged CM) of Fig. 2 definitely outperforms the CM of Fig. 1. However, in the presence of deviations close to the unity, the state-of-the-art CM remains the best. In this case, the proposed countermeasures still leaks less that an

[2] This approach clearly differs from that put forward in [6, §5.2] for comparing univariate side channel attacks (treated in the special $d = 1$ case, *i.e.* the linear case). In the later paper, the coefficients are drawn at random in the $[-1, +1]$ interval, irrespective of the sensitive data (*i.e.* the model is randomized, of expectation a "null model"), whereas in our paper, the coefficients are considered as deviations from a known non-trivial model.

unprotected design. Nonetheless, we insist that this situation is unlikely, as the deviations from the assumed Hamming distance model is of the order of one bit flip. This means that the designer has a very poor knowledge of the technology as he applies the countermeasure without checking the assumption (Property 1).

Eventually, it is noteworthy that state-of-the-art CM is even slightly improved by the imperfection of the leakage function A. This reflects the fact that the random variable $\mathsf{HW}(Z \oplus M_1) + \mathsf{HW}(M_1)$ do carry a lot of information on Z, and the noise help reduce the dependency (and thus favors the defender).

Also, both CM are equivalent for an intermediate deviation of 0.5. As this value is already quite large, we can conclude that our countermeasure is relevant even if the assumptions on the hardware leakage are extremely approximate.

6 Conclusion and Perspectives

We have presented a new masking scheme for hardware sbox implementations. We have argued that our proposal is a leak-free countermeasure under some realistic assumptions about the device architecture. The solution has been evaluated within an information-theoretic study, proving its security against 1O-SCA under the Hamming distance assumption. When the leakage function deviates slightly from this assumption (by a few tens of percent), our solution still achieves excellent results. However, if the model is very noisy (the model deviates from the Hamming distance by $\approx 50\%$), then our countermeasure remains all the same as good as state-of-the-art countermeasures.

It has been underlined (in the second construction) that some functions F have a balanced derivative in more than one direction $\alpha \neq 0$. As a perspective, we mention that this feature can be taken advantage of to increase the security of the countermeasure. Indeed, in the perfect model, the leakage remains null. However, using many αs certainly help counter model imperfections, thus reducing the leakage in this case.

Also, we underline that the proposed countermeasure can be adapted to the hypothetical case where the perfect model is not the Hamming distance $A(X, Y) = \mathsf{HW}(X \oplus Y)$, but is asymmetrical in rising and falling edges (*e.g.* $A(X, Y) = \mathsf{HW}(X \cdot \neg Y)$). Such leakages can be found in near-field electromagnetic measurements (refer to: [13] or [20, Fig. 4, left]).

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM Side-Channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. Akkar, M.L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
3. Brier, É., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)

4. Carlet, C.: Vectorial Boolean Functions for Cryptography (June 1 2008); Crama, Y., Hammer, P. (eds.): To appear as a chapter of the volume Boolean Methods and Models. Published by Cambridge University Press
5. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–540. Springer, Heidelberg (1999)
6. Doget, J., Prouff, E., Rivain, M., Standaert, F.X.: Univariate side channel attacks and leakage modeling. J. Cryptographic Engineering 1(2), 123–144 (2011)
7. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 221–234. Springer, Heidelberg (2010)
8. Goubin, L., Patarin, J.: DES and Differential Power Analysis. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
9. Kocher, P.C., Jaffe, J., Jun, B.: 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), http://www.cryptography.com/timingattack/paper.html
10. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
11. Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
12. Peeters, É., Standaert, F.X., Donckers, N., Quisquater, J.J.: Improved Higher-Order Side-Channel Attacks With FPGA Experiments. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 309–323. Springer, Heidelberg (2005)
13. Peeters, É., Standaert, F.X., Quisquater, J.J.: Power and electromagnetic analysis: Improved model, consequences and comparisons. Integration, The VLSI Journal, special issue on Embedded Cryptographic Hardware 40, 52–60 (2007), doi:10.1016/j.vlsi.2005.12.0 13
14. Prouff, E., Giraud, C., Aumônier, S.: Provably Secure S-Box Implementation Based on Fourier Transform. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 216–230. Springer, Heidelberg (2006)
15. Prouff, E., Rivain, M.: A Generic Method for Secure SBox Implementation. In: Kim, S., Yung, M., Lee, H.W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 227–244. Springer, Heidelberg (2008)
16. Prouff, E., Rivain, M., Bevan, R.: Statistical Analysis of Second Order Differential Power Analysis. IEEE Trans. Computers 58(6), 799–811 (2009)
17. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
18. Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R., Rohatgi, P.: Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 171–184. Springer, Heidelberg (2001)
19. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
20. Sauvage, L., Guilley, S., Danger, J.L., Mathieu, Y., Nassar, M.: Successful Attack on an FPGA-based WDDL DES Cryptoprocessor Without Place and Route Constraints. In: DATE, pp. 640–645. IEEE Computer Society, Nice (2009)

21. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
22. Shah, S., Velegalati, R., Kaps, J.P., Hwang, D.: Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs. In: Prasanna, V.K., Becker, J., Cumplido, R. (eds.) ReConFig, pp. 274–279. IEEE Computer Society (2010)
23. Standaert, F.X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
24. Standaert, F.X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World Is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010), http://www.dice.ucl.ac.be/~fstandae/PUBLIS/88.pdf
25. Standaert, F.X., Rouvroy, G., Quisquater, J.J.: FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks. In: Proceedings of FPL 2006. IEEE, Madrid (2006)
26. Trichina, E.: Combinational logic design for aes subbytes transformation on masked data (2003), http://eprint.iacr.org/2003/236, not published elsewhere. e.v.trichina@samsung.com 12368 (received November 11, 2003)
27. Veyrat-Charvillon, N., Standaert, F.X.: Mutual Information Analysis: How, When and Why? In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 429–443. Springer, Heidelberg (2009)
28. Waddle, J., Wagner, D.: Towards Efficient Second-Order Power Analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)

A Simulation Results in the Imperfect Model

We assume that F has been designed thanks to the first construction presented in Sec. 4.1. Hence it is a function from $\mathbb{F}_2^{n+1}$ into $\mathbb{F}_2^n$. The mask M and the constant α are of dimension $n+1$, whereas Z is n-bit long.

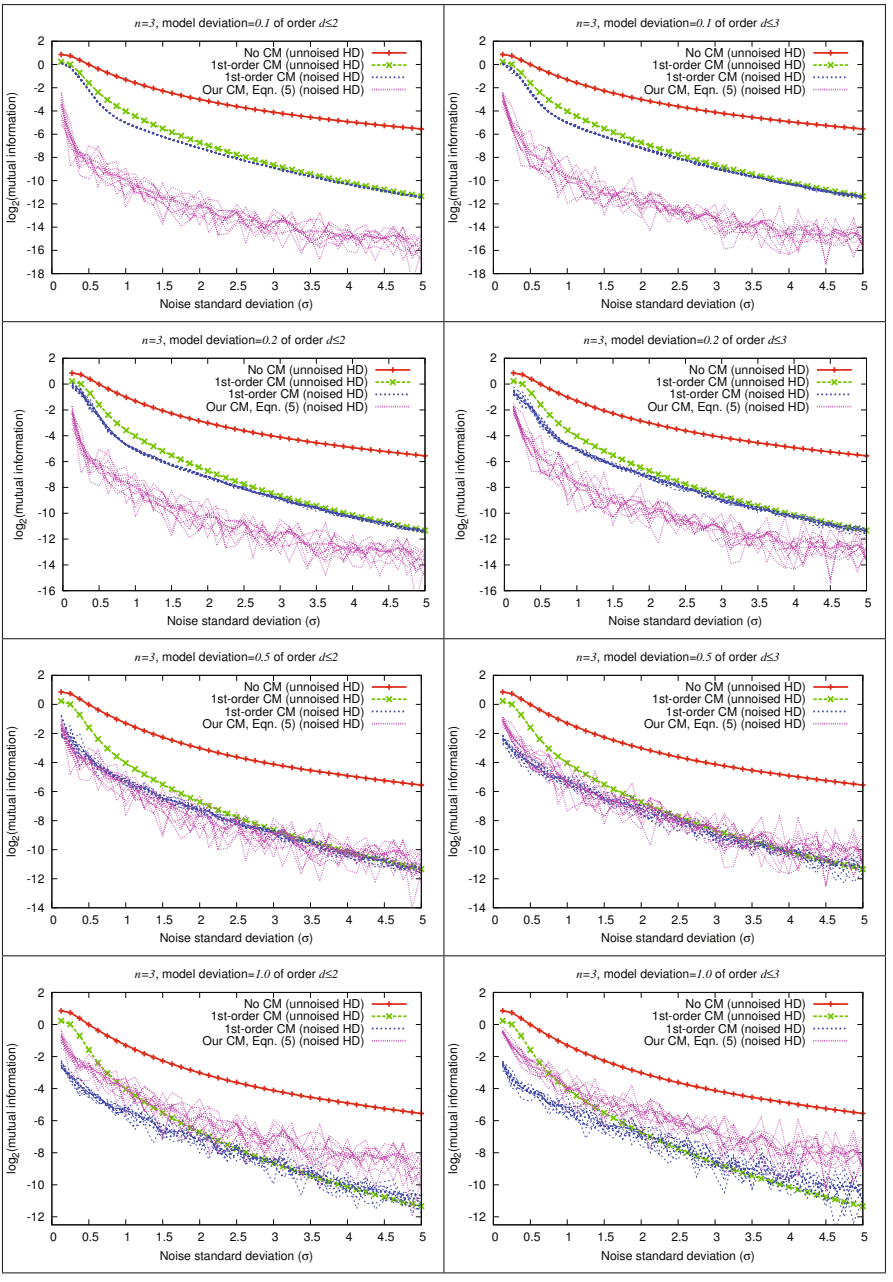
The mutual information $\mathsf{I}[O+N; Z, Z']$ is represented in Tab. 1 for:

- a Gaussian noise N of standard deviation σ varying in $]0, 5]$,
- $n = 3$ bit (to speed up the computations),
- $E = \{0\} \times \mathbb{F}_2^n \subset \mathbb{F}_2^{n+1}$ and the constant α is equal to 1000 in binary, and
- $F(x_3x_2x_1x_0) = 0$ if $x_3 = 1$ or $x_2x_1x_0$ otherwise.

For each experiment just described, we also compute the mutual information for the straightforward CM of the state-of-the-art (implementation of [25] represented in Fig. 1). We also give the mutual information of this CM if the model is exactly the Hamming distance, and indicate the corresponding leakage without any countermeasure. We recall that, still with a perfect model, the mutual information for our countermeasure with (Z, Z') is null, whatever sigma.

For every $d \in \{2, 3\}$ and deviation $\in \{0.1, 0.2, 0.5, 1.0\}$, the random number generator is seeded the same. The noisy Hamming distance model is plotted for ten sets of random coefficients $a_{(u,v)}$ defined in (7), and the average is superimposed using a thick line.

Table 1. Leakage comparison of one state-of-the-art CM and our proposed CM in the imperfect model



Practical Realisation and Elimination of an ECC-Related Software Bug Attack

Billy B. Brumley[1], Manuel Barbosa[2], Dan Page[3], and Frederik Vercauteren[4]

[1] Department of Information and Computer Science,
Aalto University School of Science, P.O. Box 15400, FI-00076 Aalto, Finland
billy.brumley@aalto.fi
[2] HASLab/INESC TEC
Universidade do Minho, Braga, Portugal
mbb@di.uminho.pt
[3] Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK
page@cs.bris.ac.uk
[4] Department of Electrical Engineering, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
fvercaut@esat.kuleuven.ac.be

Abstract. We analyse and exploit implementation features in OpenSSL version 0.9.8*g* which permit an attack against ECDH-based functionality. The attack, although more general, can recover the entire (static) private key from an associated SSL server via 633 adaptive queries when the NIST curve P-256 is used. One can view it as a software-oriented analogue of the bug attack concept due to Biham et al. and, consequently, as the first bug attack to be successfully applied against a real-world system. In addition to the attack and a posteriori countermeasures, we show that formal verification, while rarely used at present, is a viable means of detecting the features which the attack hinges on. Based on the security implications of the attack and the extra justification posed by the possibility of intentionally incorrect implementations in collaborative software development, we conclude that applying and extending the coverage of formal verification to augment existing test strategies for OpenSSL-like software should be deemed a worthwhile, long-term challenge.

Keywords: Elliptic curve, OpenSSL, NIST, fault attack, bug attack.

1 Introduction

Concrete implementation of cryptographic primitives is becoming easier as a result of more mature techniques and literature. Elliptic Curve Cryptography (ECC) is a case in point: twenty years ago ECC was limited to experts, but is now routinely taught in undergraduate engineering courses. However, such implementation tasks are still hugely challenging. This is because as well as functional correctness, the quality of an implementation is, in part, dictated by efficiency (e.g., execution speed and memory footprint) and physical security.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 171–186, 2012.

For (at least) two reasons, the efficiency of cryptographic primitives is an important issue within many applications. On one hand, many primitives represent an inherently expensive workload comprised of computationally-bound, highly numeric kernels. On the other hand, said primitives are often required in high-volume or high-throughput applications; examples include encryption of VPN traffic and full-disk encryption, both of which represent vital components in e-business. Both reasons are amplified because the primitive in question will often represent pure overhead at the application level. That is, cryptography is often an implicit enabling technology rather than an explicit feature: there is evidence to show it is common (and perhaps sane [11]) for users to disable security features in an application if it improves performance or responsiveness.

To summarise, some engineer must find an efficient way to map a complex, high-level specification of some primitive onto the characteristics of a demanding target platform, potentially using low-level programming languages and tools. Both the semantic gap between specification and implementation, and the skills gap between cryptography and engineering can be problematic. Two examples of the problems encountered, both relating to components in modern e-business work-flows, are as follows:

1. Nguyen [14] described an attack on GPG version 1.2.3, an open-source implementation of the OpenPGP standard. In short, the size of some security-critical parameters had been reduced; this meant computation was faster, but that the system as a whole was vulnerable to attack.
2. In part because of such wide-spread use, the open-source OpenSSL library has been subject to numerous attacks. Examples include issues relating to random number generation[1], and badly formulated control-flow logic allowing malformed signatures to be reported as valid[2].

Although other factors clearly contribute, one could argue that overly zealous optimisation is a central theme in both cases. Focusing on the provision of ECC in OpenSSL version 0.9.8*g*, this paper presents further evidence along similar lines. We stress that our aim is not to implicitly or explicitly devalue OpenSSL: one can, and *should*, read the paper more as a case study on the difficulty of cryptographic software implementation.

At the crux is an arithmetic bug, initially reported on the `openssl-dev` mailing list [16] in 2007 and later traced to the modular arithmetic underlying implementation of specific NIST elliptic curves; in short, the bug causes modular multiplications to (transiently) produce incorrect output. To the best of our knowledge, no cryptanalytic exploitation of this bug was previously known. Perhaps for this reason, it has not been considered a security risk, but rather a minor issue of functionality. Indeed, although the bug has been resolved in OpenSSL versions 0.9.8*h* and later it persists[3]; for example versions of the library are deployed in (at least) two major Linux distributions, namely Debian (as late as 5.0 "Lenny") and Ubuntu (as late as 9.10 "Karmic").

[1] `http://www.openssl.org/news/secadv_20071129.txt`
[2] `http://www.openssl.org/news/secadv_20090107.txt`
[3] `http://marc.info/?t=131401133400002`

The main contribution of this paper is a concrete attack: we show how the bug can be exploited to mount a full key recovery attack against implementations of Elliptic Curve Diffie-Hellman (ECDH) key agreement. The nature of the bug means the attack represents a software analogue (or a first practical realisation) of the bug attack concept [4] due to Biham et. al. Our attack works whenever the ECDH public key is static, and therefore reused across several key agreement protocol executions. In particular, any higher-level application relying on the SSL/TLS implementation of OpenSSL in the following two scenarios could be vulnerable:

1. Use of *static* ECDH-based cipher suites, (e.g., ECDH-ECDSA and ECDH-RSA). In such cipher suites, the TLS server holds a public key certificate that directly authenticates the ECDH public key; this is shared across an arbitrary number of key exchanges.
2. Use of *ephemeral* ECDH-based cipher suites (e.g., ECDHE-ECDSA and ECDHE-RSA) in combination with the OpenSSL *ephemeral-static* ECDH optimisation. In such cipher suites, and according to the TLS specification, a fresh ECDH public key should be generated for each key exchange. However OpenSSL allows one-time generation of said key when the TLS server is initialised, sharing it across an arbitrary number of key exchanges thereafter.

As a concrete example, we demonstrate the attack on stunnel version 4.42 (when linked against OpenSSL version 0.9.8*g*), an SSL-based tunnelling proxy.

As well as discussing potential countermeasures for vulnerable versions of the library, we also explore an alternative, longer-term solution. Specifically, we investigate use of formal verification as a means to *prevent* similar bugs rather than just detecting them a posteriori. This approach is particularly relevant in addressing the possibility of *intentionally* incorrect implementations, which could constitute a serious risk in software components developed using an open, collaborative approach. Our conclusion is that although such techniques can already play an important role, a step-change in attitude toward their use is required as software complexity increases; despite the effort required to adopt a development strategy that supports formal verification, this seems an important area for future work in the context of OpenSSL-like software.

From here on we use OpenSSL as a synonym for OpenSSL version 0.9.8*g* unless otherwise stated. In Section 2 we present a detailed analysis of features in OpenSSL that permit our attack to work, before explaining the attack itself, and possible countermeasures, in Section 3. In Section 4 we discuss approaches to formal verification that could help prevent similar defects in OpenSSL, and therefore similar attacks, and offer some concluding remarks in Section 5.

2 Background and Analysis

The aim of this section is to relate high-level, standardised ECC with an analysis of associated low-level implementation features in OpenSSL which support our attack.

2.1 OpenSSL Implementation of NIST Standard Curves

For a GM-prime p, multi-precision integer multiplication modulo p can be particularly efficient. OpenSSL uses this fact to support ECC implementations over the NIST standard curves P-192, P-224, P-256, P-384 and P-521. Using P-256 as an example, we have

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

and, from here on, we refer to the resulting elliptic curve as E.

Assuming a processor with a 32-bit word size, imagine that given two 8-word operands $0 \leq x, y < p$, the goal is to compute $x \cdot y \pmod{p}$. Solinas demonstrates [17, Example 3, Page 20] that given $z = x \cdot y$, the 16-word integer product of x and y, one can compute $z \pmod{p}$ by first forming nine 8-word intermediate values

$$\begin{aligned}
S_0 &= (\ z_7,\ z_6,\ z_5,\ z_4,\ z_3,\ z_2,\ z_1,\ z_0\ ) \\
S_1 &= (\ z_{15}, z_{14}, z_{13}, z_{12}, z_{11},\ 0,\ 0,\ 0\ ) \\
S_2 &= (\ 0,\ z_{15}, z_{14}, z_{13}, z_{12},\ 0,\ 0,\ 0\ ) \\
S_3 &= (\ z_{15}, z_{14},\ 0,\ 0,\ 0,\ z_{10},\ z_9,\ z_8\ ) \\
S_4 &= (\ z_8,\ z_{13}, z_{15}, z_{14}, z_{13}, z_{11}, z_{10},\ z_9\ ) \\
S_5 &= (\ z_{10},\ z_8,\ 0,\ 0,\ 0,\ z_{13}, z_{12}, z_{11}\ ) \\
S_6 &= (\ z_{11},\ z_9,\ 0,\ 0,\ z_{15}, z_{14}, z_{13}, z_{12}\ ) \\
S_7 &= (\ z_{12},\ 0,\ z_{10},\ z_9,\ z_8,\ z_{15}, z_{14}, z_{13}\ ) \\
S_8 &= (\ z_{13},\ 0,\ z_{11}, z_{10},\ z_9,\ 0,\ z_{15}, z_{14}\ )
\end{aligned}$$

and then computing $S \pmod{p}$ with

$$S = S_0 + 2S_1 + 2S_2 + S_3 + S_4 - S_5 - S_6 - S_7 - S_8\,. \tag{1}$$

Note that $|S|$ cannot be much larger than p, meaning a small number of extra modular additions or subtractions, depending on the sign of S, would give the correct result.

OpenSSL adopts a similar approach for P-192, P-224 and P-521 but deviates for P-256 and P-384: we again use P-256 as an example, but note that the same problem exists for P-384. It proceeds using the following faulty algorithm: first it computes $t = S \bmod 2^{256}$ and the correct carry c (which is positive or negative) such that

$$S = t + c \cdot 2^{256}.$$

Note that per the comment above, the carry has a small magnitude; by inspection it is loosely bounded by $-4 \leq c \leq 6$, which is used from here on wlog. The result is computed, potentially incorrectly, via two steps:

1. set $r' = (t - c \cdot p) \bmod 2^{256}$, then
2. if $r' \geq p$, $r' = r' - p$.

The concrete implementation of these steps uses a fixed look-up table $T[i] = i \cdot p \bmod 2^{256}$ for small i, by computing $r' = t - \mathrm{sign}(c) \cdot T[|c|] \bmod 2^{256}$. The modular

reduction in this case is implicit, realised by truncating the result to 8 words. The intention is to eliminate any possibility of overflow; the assumption is that c is the exact quotient of division of S by p.

The reasoning behind the faulty algorithm is that if one writes $S = t + c \cdot 2^{256}$, then the exact quotient $q = S \div p$ is given by

1. if $c \geq 0$, then $q = c$ or $q = c + 1$,
2. if $c < 0$, then $q = c$ or $q = c - 1$

since c is small. Indeed, write $\Delta = 2^{256} - p$, then after subtracting $c \cdot p$ we obtain

$$S - c \cdot p = t + c \cdot 2^{256} - c \cdot p = t + c \cdot \Delta \,.$$

Since $-4 \leq c \leq 6$ and $\Delta < 2^{224}$, this shows the result is bounded by $-p < t + c \cdot \Delta < 2p$. The faulty algorithm therefore computes an incorrect result in the following cases:

- If $c \geq 0$, the algorithm fails when $t + c \cdot \Delta \geq 2^{256}$ since it computes r' only modulo 2^{256} and not as a full integer (for which the resulting algorithm would have been correct). Note that in this case the correct result would be $r' + \Delta$ and that modulo p, the correct result thus is $r' + 2^{256} \pmod{p}$.
- If $c < 0$, the algorithm fails when $t + c \cdot \Delta < 0$. The correct result then depends on whether $(t + c \cdot \Delta) \bmod 2^{256} \geq p$ or not: in the former case, the correct result is $r' - \Delta$, whereas in the latter case, the correct result is given by $r' + 2^{256} - 2\Delta$. Note that although there are two different subcases for $c < 0$, the errors $-\Delta$ and $2^{256} - 2\Delta$ are congruent modulo p, i.e. modulo p, the correct result is given by $r' - 2^{256} \pmod{p}$.

Note that Ciet and Joye [5, Section 3.2] consider the case of faults in the underlying field; the fault (resp. bug) here is certainly related, but occurs as a result of erroneous computation rather than corrupted parameters.

The resulting bug is difficult to detect using the (random) test vector approach employed by OpenSSL: it simply manifests itself too rarely. An upper bound for the probability of the bug being triggered can be obtained by ignoring the probabilities of certain carries occurring and analysing the case $t + 6 \cdot \Delta \geq 2^{256}$: if t was chosen uniformly over the interval $[0, 2^{256}[$, then this case occurs with probability less than 2^{-29}, so OpenSSL computes the result incorrectly with probability less than $10 \cdot 2^{-29}$.

To *deliberately* trigger the bug, we empirically arrived at the following strategies (after inspection of partial products within the integer product of x and y):

- For modular multiplication, selecting x, y as follows should induce an incorrect result for any random $0 \leq R_x, R_y < 2^{31}$:

$$\begin{aligned} x &= (2^{32} - 1) \cdot 2^{224} + 3 \cdot 2^{128} + R_x \\ y &= (2^{32} - 1) \cdot 2^{224} + 1 \cdot 2^{96} \ + R_y \end{aligned}$$

- For modular squaring, selecting $x = (2^{32} - 1) \cdot 2^{224} + 1 \cdot 2^{129} + R_x$, should induce an incorrect result for any random $0 \leq R_x < 2^{31}$.

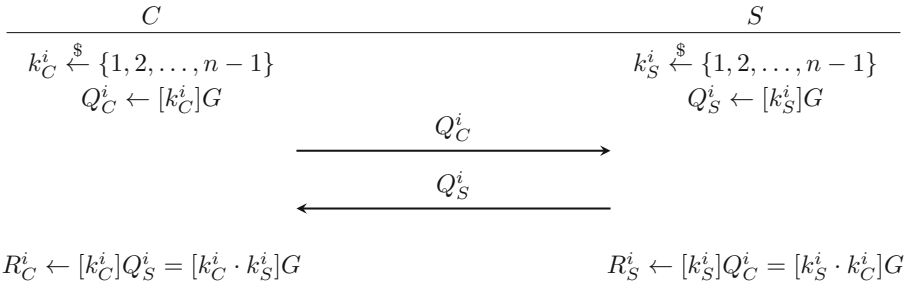


Fig. 1. A description of ECDH key exchange

2.2 ECC Cipher Suites for TLS

Modern versions[4] of the Transport Layer Security (TLS) standard provide a number of different cipher suites that rely on ECC for key exchange. We focus on Elliptic Curve Diffie-Hellman (ECDH) and Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) based cipher suites[5]. To be precise, in these cipher suites the key exchange protocol is conducted to establish a secret key for a session i between a client C and a server S; it proceeds in three stages outlined in Figure 1, with client and server assumed to share $D = \{p, A, B, x_G, y_G, n, h\}$, a set of domain parameters. After the protocol terminates, $R_S^i = R_C^i$ represents the shared key.

Figure 2 illustrates the TLS handshake at a higher level of abstraction. We now describe how said handshake proceeds, detailing how the ECDH protocol messages formalised above are embedded in the communication. While our attacks are equally applicable in the case of client authentication, we omit the details for this case. The `ClientHello` message conveys the protocol version, supported cipher and compression methods, and a nonce to ensure freshness. The `ServerHello` message is analogous, but selects parameters from the methods proposed by the client (contingent on the server supporting them). The content of the `Certificate` message varies depending on the selected cipher suite:

- In ECDH-ECDSA, the `Certificate` message contains a static ECDH public key authenticated by a public key certificate signed with ECDSA; ECDH-RSA is analogous, but the public-key certificate is signed using an RSA signature. The static ECDH public key corresponds to the value Q_S^i above, and the server will reuse this value for multiple key exchanges with an arbitrary number of clients. For this reason, the `ServerKeyExchange` message is omitted in ECDH suites.

[4] http://tools.ietf.org/html/rfc5246
[5] http://tools.ietf.org/html/rfc4492

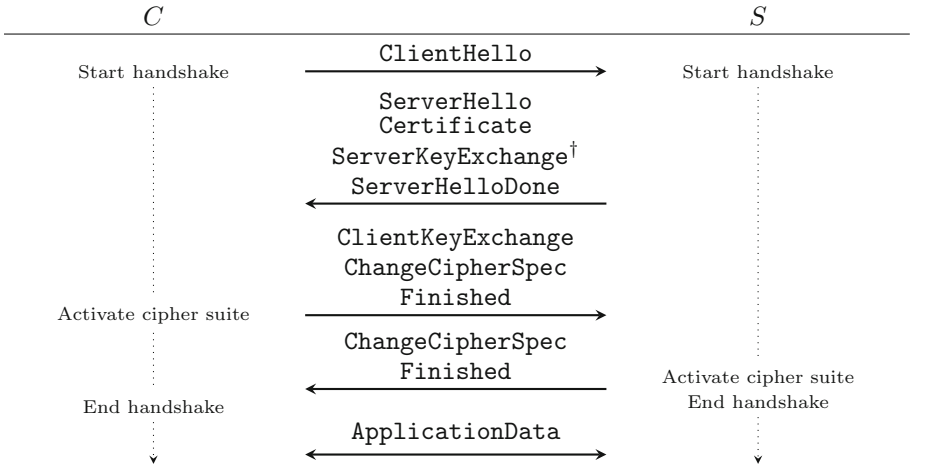


Fig. 2. Message flow in a TLS handshake with ECDH cipher suites; messages relating to client authentication are omitted for clarity, and those marked with † are only sent under specific circumstances

– In ECDHE-ECDSA, the `Certificate` message contains an ECDSA verification key, which is authenticated by a certificate signed with the same algorithm; ECDHE-RSA is analogous but an RSA signature verification key is sent, and the public-key certificate is signed using an RSA signature. The server also sends message `ServerKeyExchange`, containing both a fresh ephemeral ECDHE public key (i.e., Q_S^i) and a digital signature authenticating this and other handshake parameters, including the exchanged nonces. Said signature is produced with either the ECDSA or RSA signing key matching to the verification key sent in the `Certificate` message.

The `ServerHelloDone` message marks the end of this stage of the protocol; the client then sends its ephemeral ECDHE key in the `ClientKeyExchange` message, which always includes a fresh ephemeral Q_C^i. Finally, the negotiated symmetric cipher suite is activated via the `ChangeCipherSpec` message. A session key for the cipher suite is derived from $R_S^i = R_C^i$ and the exchanged nonces. The `Finished` messages provide key confirmation for both parties, notably occurring client-first, and depend on all previous messages in the protocol execution.

2.3 OpenSSL Implementation of the ECC Cipher Suites

The ECDH implementation in OpenSSL is seemingly straightforward, and follows the TLS specification. However, the ECDHE implementation offers two distinct options for server applications. The first follows the specification and generates a new ephemeral ECDH key pair for every protocol execution. Deviating from

the specification, the second features an optimisation termed ephemeral-static ECDH[6].

When activated, the optimisation means a single ECDH key pair is generated during initialisation of the OpenSSL context within an application. This key pair is reused for all protocol executions thereafter; with ephemeral-static ECDH, OpenSSL has the server use a static key (i.e., a fixed k_i^S and hence Q_i^S for all i) for each OpenSSL context. Put another way, the key pair is ephemeral for each application instance and not (necessarily) per handshake instance. While this preserves forward secrecy between application instances, it violates forward secrecy within a single application instance when performing more than a single protocol execution. Interestingly, the default behaviour is the latter: to "opt out" and disable the optimisation, the application must explicitly use the `SSL_OP_SINGLE_ECDH_USE` option during initialisation of the context.

3 An Attack on ECDH in OpenSSL

Implementing Scalar Multiplication. For scalar multiplication on $E(\mathbb{F}_p)$, OpenSSL uses a textbook double-and-add algorithm along with the modified width-w NAF representation of k. For P-256 OpenSSL sets $w = 4$, i.e., each non-zero digit from digit set $\mathcal{D} = \{0, \pm 1, \pm 3, \pm 5, \pm 7\}$ is followed by at least three zero digits. Modified NAF is otherwise identical to traditional NAF but allows the most-significant digit to violate the non-adjacency property, if doing so does not increase the weight but reduces the length. This slight distinction between the two affects neither the derivation of our attack nor the implementation of it: henceforth we use NAF synonymously with traditional NAF.

Attack Goals and Limitations. The goal of the attacker is to recover the fixed k_S in the server-side computation of $R_S^i = [k_S]Q_C^i$. The algorithm we propose and implement recovers the digits of k_S by detecting faults in the server-side computation of R_S^i. The ability of the attacker to observe these faults heavily depends on the protocol and/or cryptosystem under attack. For example, when k_S is fixed but Q_C^i is not, it is uncommon for a protocol to directly reveal R_S^i to another participant. Inspecting TLS, one way the attacker can detect faults is by attempting to complete the handshake. If R_S^i is fault-free (denoted $R_S^i \in E$) then the computed session key is the same for both the client and server. Consider the phase of the handshake when the client sends `ChangeCipherSpec`, signalling that it has activated the newly negotiated session cipher suite, and transmits the encrypted `Finished` handshake message for key confirmation. Then, if the server successfully decrypts said message, the handshake continues, and ultimately succeeds. On the other hand, if R_S^i is faulty (denoted $R_S^i \notin E$) then the session keys differ, the server will not obtain the expected key confirmation message, and the handshake ultimately fails. The practical consequence is that the attacker *cannot* arbitrarily choose each Q_C^i in the protocol, rather he *must* know the discrete logarithm of said point in order to correctly calculate the negotiated session key.

[6] `http://tools.ietf.org/html/rfc5753`

The Attack Algorithm. Having established a method to detect the occurrence of faults, the algorithm proceeds in an exhaustive depth-first search for NAF(k_S) starting from the most-significant digit, trimming limbs and backtracking by iteratively observing handshake results. At each node in the search, the attacker submits a different carefully chosen point, termed a *distinguisher point*, to determine if the next unknown digit takes a specific value at the given iteration, tracing the execution path of the server-side scalar multiplication. We define a distinguisher point for an integer prefix a and target digit $b \in \mathcal{D} \setminus \{0\}$ to be a point $D_{a,b} = [l]G \in E$ such that $[a \parallel b \parallel d]D_{a,b} \notin E$ and $[a \parallel c \parallel d]D_{a,b} \in E$ for all $c \in \mathcal{D} \setminus \{0, b\}$ both hold. Here, l is known, a is the known portion of NAF(k_S), and d is any sufficiently long random padding string that completes the resulting concatenation to a valid NAF string. In practice, testing a single distinguisher point requires varying d over many values to ensure the computation reaches a sufficient number of possible subsequent algorithm states: this acts to deter false positives.

We step through the first few iterations to demonstrate the algorithm. For clarity, we use subscripts on variables a and $\mathcal{D}$ to identify the iteration, i.e., the digit index in the NAF representation from most- to least-significant, we are referring to. For $i = 1$, a_1 is the empty string and $\mathcal{D}_1 = \{1, 3, 5, 7\}$. The attacker finds a distinguisher point $D_{\emptyset,b}$ for each $b \in \mathcal{D}_1 \setminus \{1\}$ and uses these three points in attempted handshakes to the server[7]. Handshake failure reveals the correct digit, and allows us to set a for the next iteration as $a_2 = b$; if all handshakes succeed, the attacker deduces $a_2 = 1$ for the next iteration. Enforcing NAF rules, for $i = 5$ we have $a_5 = a_2 \parallel 000$ and $\mathcal{D}_5 = \{0, \pm 1, \pm 3, \pm 5, \pm 7\}$. The attacker then finds $D_{a_5,b}$ for each $b \in \mathcal{D}_5 \setminus \{0\}$ and uses these eight points in attempted handshakes to the server. Handshake failure reveals the correct $a_6 = a_5 \parallel b$ and if all handshakes succeed the attacker deduces $a_6 = a_5 \parallel 0$. The attack continues in this manner to recover all subsequent digits. On average, our attack takes 4 handshake attempts to recover subsequent non-zero digits, and 8 handshake attempts to detect zero digits (note that we do not explicitly check for zeros which are implied by the NAF representation).

Relation to the Bug Attacks of Biham et al. This algorithm relates to that which Biham et al. used to mount a bug attack against Pohlig-Hellman in the $\mathbb{F}_p^*$ setting [4, Section 4.1.1]. The authors consider recovering the binary digits of the exponent from a left-to-right binary modular exponentiation algorithm. It does this by finding input $X \in \mathbb{F}_p^*$ such that $(X^2)^2$ fails yet $(X^2)X$ does not, i.e., it uses the former to query explicitly for any zero digits and implicitly obtain any non-zero digits. Assume the attacker knows l such that $X = g^l$ (this is not necessary to carry out their attacks, but is necessary when adapting their strategy to attack TLS). The most-significant digit of the victim's binary string is a non-zero digit by definition. The attacker queries the next digit by submitting X. Assume wlog. that it fails: the attacker now knows the two most-significant

[7] When $i = 1$ finding a distinguisher point $D_{\emptyset,1}$ is less practical as it can cause the table of pre-computed points to be erroneously populated, so in this case querying for that particular digit value occurs implicitly.

digits are 1 and 0. To obtain the (or analogy of a) distinguisher point for the next iteration, the attacker simply computes $X^{1/2}$ and, knowing the discrete logarithm of X to the base g, can easily derive the logarithm of this group element. This procedure essentially cancels out the effect of the known digits, forcing the accumulator to take value X at the intended iteration.

In the discussion that follows we will see that, in our attack, we search for all distinguishing points independently. Indeed, although it is tempting to use existing distinguisher points to derive subsequent points, which would reduce the complexity of our attack, this approach does not seem to apply in our scenario. The distinguishing point derivation technique by Biham et al. works for $\mathbb{F}_p^*$ (and even $E(\mathbb{F}_p)$ when using affine coordinates) because there is a unique representation for group elements in both cases (i.e., elements of $\mathbb{F}_p$ are stored as their smallest non-negative residue). There are a number of reasons why this strategy is ineffective for the OpenSSL implementation of ECC, the most prominent being the use of projective coordinates: group element representations are not unique, as the implementation computes in an equivalence class. As a result, the attacker cannot force the accumulator to the desired value since it undergoes projective point doublings. To summarise, there is no obvious way to cancel out the effect of the known digits. In any case, we will see in the following that this does not impact on the practicality of our attack in a significant way.

Finding Distinguisher Points. Lacking an analytical method to derive distinguisher points, our implementation of the attack resorts to random search. This is done by choosing l at random and testing whether $D_{a,b}$ satisfies the properties of a distinguisher point and, as previously mentioned, varying d over a reasonable amount of random values. The practicality of carrying out the attack thus hinges (in part) on the effort required to find said points, i.e., the average number of l values to test. Figure 3 (left) depicts our observed effort in two different parts of the attack. The solid line represents effort required at the beginning of the attack with less than 12 digits recovered. The dashed line represents effort required towards the end of the attack with roughly 224 digits recovered. This empirical data suggests not only that the computational effort to find a specific distinguisher point is rather modest, but that said effort does not significantly depend on the amount of known key digits. That is, as the attacker recovers successive digits, the effort to find new distinguisher points remains fairly constant. It is worth mentioning that the search scales perfectly with the number of computing nodes.

Attack Analysis and Results. The practicality of the attack additionally hinges on the number of required distinguisher points, i.e., the average number of attempted handshakes to recover the entire key. Our theoretical analysis of the expected number of queries, based on a rough approximation to the distribution of digits in the NAF representation of a random 256-bit secret exponent, points to an approximate value of 635 handshakes, suggesting that said value is similarly modest. We also measured this value empirically and obtained a consistent result, as illustrated in Figure 3 (right).

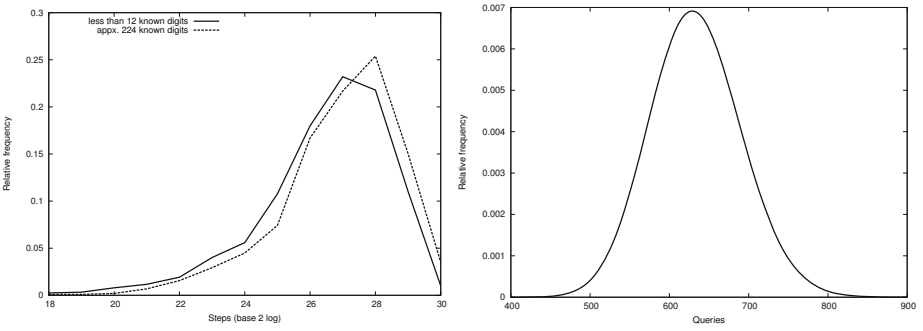


Fig. 3. Left: distribution of required search steps to find distinguisher points (lg-linear). The solid line represents effort towards the beginning of the attack (mean 26.5 s.d. 2.0) and the dashed line towards the end (mean 26.9 s.d. 1.9). Right: distribution of required queries to the server, or distinguisher points, for the full attack (mean 633.2 s.d. 57.7).

The proof-of-concept source code for our attack implementation includes distinguisher points for all NAF strings up to length 12. As it stands this immediately removes roughly 12 bits of entropy from the private key, and is of course easily extendible. The code includes instructions for running the attack in two different use cases:

1. The stunnel application provides a flexible SSL proxy for any application that does not natively support SSL; it links against OpenSSL to provide the SSL functionality. When stunnel is configured to support ECDH suites with a static ECDH key, our attack implementation facilitates recovery of said private key. Once carried out, it allows the attacker to decrypt all previous SSL sessions and to impersonate the server indefinitely.
2. The s_server application within OpenSSL is a generic SSL server. This application, and those similar to it supporting ECDHE suites, are vulnerable since they feature the ephemeral-static ECDH optimisation. The attack implementation facilitates recovery of the application instance's ECDH private key. Once carried out, it allows the attacker to decrypt all previous SSL sessions from the application instance and to impersonate the server until the application restarts.

Algebraic and Algorithmic Countermeasures. Coron outlines three methods to thwart DPA attacks [7, Section 5]. In general, they seek to counteract the deterministic nature of double-and-add style scalar multiplication routines, and are therefore quite effective against the bug attack presented above.

Scalar blinding, i.e., $[k]P = [k + rn]P$ for (small) random value r, effectively randomises the execution path of the scalar multiplication algorithm. This is not "free" however: the performance overhead (and security) is proportional to the size of the random multiplier. Point blinding, i.e., $[k]P = [k](P + R) - S$, with randomly chosen $R \in E$ and $S = [k]R$ (updating both $R = [r]R$ and $S = [r]S$ for small, random r periodically), is equally effective. However, this also entails

some performance overhead and is slightly more intrusive to integrate. Lastly, coordinate blinding, i.e., multiplying the projective coordinates of the accumulator by a random element of the finite field in such a way that preserves the equivalence class, effectively randomises the states of the scalar multiplication algorithm. In this case, said blinding would only need to occur at the beginning of the algorithm and hence does not entail anywhere near as significant a performance overhead. Our implementation as a patch to the OpenSSL source code is available from the `openssl-dev` mailinglist[8].

Algorithmic countermeasures seem ineffective against the attack if they are deterministic. The Montgomery ladder [13, Section 10.3.1], for example, can resist many forms of side-channel attack due to the regular nature of operations performed; it cannot resist a varient of the attack in Section 3 however, since one can still select distinguisher points that target the control-flow and hence (iteratively) recover the scalar. See the full version of this paper for a more detailed discussion.

4 Approaches to Formal Verification

In this section we investigate whether it is realistic to use current formal verification technology to prevent similar bug attacks in open-source projects such as OpenSSL. We focus our analysis in two complementary aspects of this problem: first the design of efficient algorithms for carrying out the necessary numeric computations, and second checking that these algorithms are correctly implemented in machine code. The concrete arithmetic bug that we explore in this paper serves as a perfect illustration of why these two aspects should be considered separately.

A high-level specification of the procedure for modular reduction that was found to be incorrect is described in Section 2.1. Producing a concrete implementation of this procedure implies a need to refine it into a lower-level specification; the particular refinement we are analysing can be described as follows:

1. Pre-compute a table T, where the i-th element $T[i] = i \cdot p$ (for small i).
2. To reduce the integer product $z = x \cdot y$ modulo p, first construct the intermediate values $S_0, S_1, \ldots, S_8$ based on the representation of z.
3. Write the integer sum $S = S_0 + 2S_1 + 2S_2 + S_3 + S_4 - S_5 - S_6 - S_7 - S_8$ as $S = t + 2^{256} \cdot c$.
4. Return the result $r' = t - \mathrm{sign}(c) \cdot T[|c|] \pmod{2^{256}}$.

This highlights a subtle point: rather than a programming error, the bug is more accurately characterised as a design error. That is, the *incorrectly designed* refinement is *correctly implemented* in OpenSSL.

We next discuss how one can formally verify the design of such refinements using existing technology. We discuss the viability of fully verifying implementation correctness in the full version of this paper.

[8] http://marc.info/?l=openssl-dev&m=131194808413635

The first question we consider is whether it is feasible to verify that a particular refinement is correct wrt. the associated high-level specification. In order to illustrate the techniques that can be employed at this level, we present two examples inspired in the bug described in the previous sections; each represents a refinement (invalid and valid respectively) along the lines above.

We have used the CAO domain specific language for cryptography [3] and the CAOVerif deductive verification tool [19]. The CAO language is syntactically close to C, but is equipped with type system (including, in particular, multi-precision integers) that make it straightforward to express mathematically-rich algorithms. The CAOVerif tool [9] takes an annotated version of the program one wishes to prove correct as input (the specification of correctness is included in the annotations), and generates a set of proof obligations that need to be validated as output. If one is able to validate all the proof obligations, this implies the program meets the specification. The proof obligations can be discharged by automatic provers such as Simplify [8] or Alt-Ergo [6] or, if these fail, one can use an interactive theorem prover such as Coq [18]. The verification condition generation procedure is based on Hoare logic [12], and uses the Jessie/Frama-C [9] and Why [10] platforms as a back-end.

Failed Proof for an Incorrect Refinement. Proving that the refinement used by OpenSSL is functionally equivalent to the original specification essentially reduces to proving that steps 1, 3 and 4 compute the same result as Equation 1. To illustrate how the proof proceeds we implemented these steps in CAO, annotating the result using the CAO Specification Language [2] (closely inspired by the ANSI C Specification Language) to indicate the proof goal. The most relevant fragment is presented below, where `Prime` and `Power2` are global variables holding the values of p and 2^{256}, respectively:

```
1  typedef modPower2 := mod[2**256];
2
3  /*@ requires ((0 <= sum) && (sum <= 7*(Power2-1))
4      && (Prime == 2**256 - 2**224 + 2**192 + 2**96 - 1)
5      && (Power2 == 2**256))
6      ensures  ((0 <= result) && (result < Prime)
7                && (exists d : int; result + d*Prime == sum)) */
8  def modPrime(sum : int) : int {
9      def c,res : int;
10     c := sum / Power2;
11     /*@ assert c >= 0 && c <= 6 */
12     /*@ assert 0 <= (sum - c*Prime) */
13     /*@ assert (sum - c*Prime) < Power2 */
14     res := (int)((modPower2)sum - (modPower2)(c*Prime));
15     if (res >= Prime) {
16        res := res - Prime;
17     }
18     return res;
19 }
```

Ignoring the embedded annotations for the moment, `modPrime` takes the summation `sum` as input (which for simplicity and wlog. we assume to be positive),

[9] A distribution of the CAOVerif tool and source code for the examples in this paper are available from `http://crypto.di.uminho.pt/CACE/`

and computes a (possibly incorrect) output of `sum` modulo `Prime`. This computation is performed in three steps that mimic the OpenSSL implementation: 1) it calculates the (computationally inexpensive) division by `Power2`, 2) it uses the result `c` to subtract a multiple of `Prime` from the input (this operation is carried out efficiently modulo `Power2` by casting the values to an appropriate data type), and 3) the result is placed in the correct range by applying a conditional subtraction.

The annotations in the code can now be described. The specification of `modPrime` is a contract including a precondition `requires` and a post-condition `ensures`. It states that provided `sum` is in the correct range, i.e., $0 \leq \texttt{sum} \leq 7 \cdot (\texttt{Power2} - 1)$, and that the output meets the mathematical definition of the least residue of `sum` modulo Prime, i.e., $(0 \leq \texttt{res} < \texttt{Prime}) \wedge (\exists d \text{ st. } \texttt{res} + d \cdot \texttt{Prime} = \texttt{sum})$. Inside the function, a series of assertions guide the proof tool toward establishing intermediate results towards an attempted proof. For example, one is able to establish the correct range of `c` after the division. The proof fails, however, when one tries to establish that performing the subsequent calculation modulo `Power2` will produce a result that is still congruent with `sum` modulo `Prime`. In particular, one will not be able to prove (if that was the initial intuition) that $\texttt{sum} - \texttt{c} \cdot \texttt{Prime} < \texttt{Power2}$ which would be sufficient to ensure that the calculations *could* be performed modulo `Power2`.

Robust Proof for a Correct Refinement. Consider the following alternative refinement to that presented above.

```
1  /*@ requires ((0 <= sum) && (sum <= 7*(Power2-1))
2      && (Prime == 2**256 - 2**224 + 2**192 + 2**96 - 1)
3      && (Power2 == 2**256))
4      ensures  ((0 <= result) && (result < Prime)
5                && (exists d : int; result + d*Prime == sum)) */
6  def modPrime(sum : int) : int {
7          def res : int := sum;
8      /*@ ghost def t : int :=0; */
9      /*@ assert res + t*Prime == sum */
10     /*@ invariant (res>=0) && (res == sum + t*Prime) */
11     while (res>=Prime) {
12         /*@ ghost t:=t-1; */
13         res := res - Prime;
14     }
15     /*@ assert ((0 <= res) && (res < Prime)
16                              && (res + (-t)*Prime == sum)) */
17     return res;
18 }
```

This implements the "natural" refinement: it simply subtracts the `Prime` from the input until the result is in the appropriate range. In order to complete the proof, one needs to include a loop invariant that keeps track of how many times `Prime` is subtracted; to achieve this, we use a "ghost" variable `t` that is only visible to the verification tool. The annotated result can be fed to the CAOVerif tool, which will automatically check that the program indeed meets the specification (noting that this automation relies partly on the assertions included).

5 Conclusions

This paper presents a concrete attack against ECDH-based functionality supported by OpenSSL version 0.9.8*g*. The attack works whenever the ECDH public key is static: this may occur either explicitly as a result of the selected cipher suite, or (partly) implicitly as a result of the (non-standard) ephemeral-static optimisation supported by OpenSSL. It is worth noting that we also considered exploiting the bug to mount invalid curve attacks [1]: while this allowed us to bypass OpenSSL point validation routines, it did not lead to a practical attack due largely to the nature of the bug severely limiting the number of invalid curves.

The arithmetic bug has been resolved in OpenSSL versions 0.9.8*h* and later. As a result, it is tempting to conclude that the attack does not represent a serious threat. However, vulnerable versions of the library are deployed in (at least) two major Linux distributions, namely Debian (as late as 5.0 "Lenny") and for Ubuntu (as late as 9.10 "Karmic"). Although they selectively apply patches to the default installation, the arithmetic bug persists in both. That is, although a patch resolving the bug has been available since 2008, it has yet to permeate existing installations. This represents a concrete example of the premise that patching is no panacea for similar problems.

Whether OpenSSL should prevent optimisations like ephemeral-static being included or invoked is perhaps a more philosophical question aligned to a bigger picture. For example, problems relating to IPsec support for encryption-only modes of the Encapsulating Security Payload (ESP) protocol seems conceptually similar; a comprehensive overview and resulting attack is given by Paterson and Yau [15]. One can conjecture that the motivation for encryption-only ESP, like ephemeral-static ECDH, is efficiency. Given that provision of a more efficient, less secure option will inevitably lead to someone using it, our work lends weight to the argument that a better approach may be to permit *only* secure, albeit less efficient options.

While (non-)support for various options in OpenSSL is subjective in part, the correctness of what *is* supported is less debatable. As such, detecting bugs in a more rigorous manner represents a difficult and extremely resource- and time-consuming task if undertaken over the entire implementation. OpenSSL clearly differs from a formal cryptographic standard, but it represents a de facto, ubiquitous and mission-critical software component in many settings. As such, we suggest that the effort required to adopt a development strategy capable of supporting formal verification is both warranted, and an increasingly important area for future work.

Acknowledgements. This work has been supported in part by EPSRC via grant EP/H001689/1 and by project SMART, funded by ENIAC Joint Undertaking (GA 120224).

References

1. Antipa, A., Brown, D.R.L., Menezes, A., Struik, R., Vanstone, S.A.: Validation of Elliptic Curve Public Keys. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 211–223. Springer, Heidelberg (2002)
2. Barbosa, M.: CACE Deliverable D5.2: formal specification language definitions and security policy extensions (2009), http://www.cace-project.eu
3. Barbosa, M., Moss, A., Page, D.: Constructive and destructive use of compilers in elliptic curve cryptography. J. Cryptology 22(2), 259–281 (2009)
4. Biham, E., Carmeli, Y., Shamir, A.: Bug Attacks. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 221–240. Springer, Heidelberg (2008)
5. Ciet, M., Joye, M.: Elliptic curve cryptosystems in the presence of permanent and transient faults. Designs, Codes and Cryptography 36(1), 33–43 (2005)
6. Conchon, S., Contejean, E., Kanig, J.: Ergo : a theorem prover for polymorphic first-order logic modulo theories (2006), http://ergo.lri.fr/papers/ergo.ps
7. Coron, J.-S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
8. Detlefs, D., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM 52(3), 365–473 (2005)
9. Filliâtre, J.-C., Marché, C.: Multi-Prover Verification of C Programs. In: Davies, J., Schulte, W., Barnett, M. (eds.) ICFEM 2004. LNCS, vol. 3308, pp. 15–29. Springer, Heidelberg (2004)
10. Filliâtre, J.-C., Marché, C.: The Why/Krakatoa/Caduceus Platform for Deductive Program Verification. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 173–177. Springer, Heidelberg (2007)
11. Herley, C.: So long, and no thanks for the externalities: The rational rejection of security advice by users. In: New Security Paradigms Workshop (NSPW), pp. 133–144 (2009)
12. Hoare, C.A.R.: An axiomatic basis for computer programming. Communications of the ACM 12, 576–580 (1969)
13. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. 48(177), 243–264 (1987)
14. Nguyen, P.Q.: Can We Trust Cryptographic Software? Cryptographic Flaws in GNU Privacy Guard v1.2.3. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 555–570. Springer, Heidelberg (2004)
15. Paterson, K.G., Yau, A.K.L.: Cryptography in Theory and Practice: The Case of Encryption in IPsec. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 12–29. Springer, Heidelberg (2006)
16. Reimann, H.: BN_nist_mod_384 gives wrong answers. openssl-dev mailing list #1593 (2007), http://marc.info/?t=119271238800004
17. Solinas, J.A.: Generalized Mersenne numbers. Technical Report CORR 99-39, Centre for Applied Cryptographic Research (CACR), University of Waterloo (1999), http://www.cacr.math.uwaterloo.ca/techreports/1999/corr99-39.pdf
18. The Coq Development Team. The Coq Proof Assistant Reference Manual – Version V8.2 (2008), http://coq.inria.fr
19. Vieira, B., Barbosa, M., Sousa Pinto, J., Filliatre, J.-C.: A deductive verification platform for cryptographic software. In: International Workshop on Foundations and Techniques for Open Source Software Certification, OpenCert (2010)

A New Pseudorandom Generator from Collision-Resistant Hash Functions

Alexandra Boldyreva* and Virendra Kumar**

School of Computer Science, Georgia Institute of Technology
266 Ferst Drive, Atlanta, GA 30332-0765 USA
{aboldyre,virendra}@cc.gatech.edu

Abstract. We present a new hash-function-based pseudorandom generator (PRG). Our PRG is reminiscent of the classical constructions iterating a function on a random seed and extracting Goldreich-Levin hardcore bits at each iteration step. The latest PRG of this type that relies on reasonable assumptions (regularity and one-wayness) is due to Haitner et al. In addition to a regular one-way function, each iteration in their "randomized iterate" scheme uses a new pairwise-independent function, whose descriptions are part of the seed of the PRG. Our construction does not use pairwise-independent functions and is thus more efficient, requiring less computation and a significantly shorter seed. Our scheme's security relies on the standard notions of collision-resistance and regularity of the underlying hash function, where the collision-resistance is required to be *exponential*. In particular, any polynomial-time adversary should have less than $2^{-n/2}$ probability of finding collisions, where n is the output size of the hash function. We later show how to relax the regularity assumption by introducing a new notion that we call *worst-case regularity*, which lower bounds the size of primages of different elements from the range (while the common regularity assumption requires all such sets to be of equal size). Unlike previous results, we provide a concrete security statement.

Keywords: Pseudorandom generator, hash function, collision-resistance, provable security.

1 Introduction

A pseudorandom generator (PRG) is an important cryptographic primitive that was introduced by Blum and Micali [3], and later formalized into its current form by Yao [23]. PRGs are used to generate *pseudo*random bits from a short random seed, which can then be used in place of truly random bits that most cryptographic schemes rely on. On the foundational side, PRGs can be used as a building block for more complex cryptographic objects like pseudorandom function (PRF) [8], bit commitment [20], etc.

* Supported in part by NSF CAREER award 0545659 and NSF Cyber Trust award 0831184.

** Supported in part by the grants of the first author.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 187–202, 2012.

In their seminal work, Håstad et al. [15] building on the previous works [17,14] show how to construct a PRG, henceforth called the HILL-PRG, from *any* one-way function. While the construction is of great theoretical value, it is extremely (orders of magnitude) inefficient compared to the Blum-Micali-Yao (BMY) PRG that builds on a one-way *permutation*. BMY-PRG is the most efficient known construction, whose security relies on a reasonable assumption. Practical standardized PRGs based on block-ciphers and hash functions (a hash function is a function whose range is smaller than the domain, also referred to as a compression function) [6], though much more efficient, rely on a rather strong and not well-studied assumption (in the theoretical cryptography community) that the underlying function is a PRF [5], and thus are not a focus of this work. In this paper, we investigate a question of finding an *efficient* hash-function-based PRG, whose security relies on *collision-resistance*, a very well-studied and widely-used property of a hash function. A collision-resistant hash function (CRHF) is of course one-way but certainly not a permutation, as it compresses the input, and hence the BMY-PRG is not suitable for our problem.

1.1 Related Work

The seed length (as a function of the input length m of the underlying function) is an important measure of the efficiency and the security of a PRG. The best known bound for the HILL-PRG of $\mathcal{O}(m^8)$ was shown by Holenstein [16]. This was later improved (for an alternative construction) to $\mathcal{O}(m^7)$ and $\mathcal{O}(m^4)$ by Haitner et al. in [11] and [13], respectively. While the efficiency is obvious from the seed length, we present an example to truly appreciate the effect of seed length on the security of a PRG. Say, we have a one-way function that is secure, according to current standards, only for inputs of size at least 128 bits, then Holenstein's proof shows that the HILL-PRG is secure only for seeds of size (ignoring constants) at least 2^{56} bits! Several works have tried to bridge this huge gap from the BMY-PRG's seed length of $\mathcal{O}(m)$, by making stronger assumptions on the underlying function. Following are the two main types of strengthening in the assumption:

- **Regularity.** Goldreich et al. [9] gave a construction of PRG with seed length $\mathcal{O}(m^3)$, whose security requires that the underlying function is one-way and *regular*. This was later improved by Haitner et al. [11], where they first present a tighter security proof for a construction similar to that of Goldreich et al., thus improving the seed length to $\mathcal{O}(m^2)$ (cf. Section 3.3 in [11]). In the following section of the same work, Haitner et al. show how the seed length can be further reduced to $\mathcal{O}(m \log m)$ by the use of bounded-space generators of Nisan [21] (or, Impagliazzo et al. [18]).
- **Exponential Hardness.** Holenstein [16] gave a construction of PRG with seed length $\mathcal{O}(m^5)$, whose security relies on the underlying function being an *exponentially hard* one-way function. This was later improved by Haitner et al. to seed length $\mathcal{O}(m^2)$ in [12] and [13], where the latter (unlike prior works) doesn't require adaptive calls to the one-way function.

1.2 Our Result

We construct a new hash-function-based PRG with seed length less than $2m$, i.e. as efficient as the BMY-PRG, thus improving the efficiency over all prior works which do not rely on permutations (i.e., function-based PRGs). Our scheme is reminiscent of the classical constructions [3,23] iterating a function on a random seed and extracting Goldreich-Levin hardcore bits [10] at each iteration step. One notable difference from the BMY-PRG is that instead of a permutation, we use a hash function. Let h be a hash function mapping strings of size m bits to strings of size n bits, for $m > n$. Assume we have a random seed $x\|r$, where both x and r are n bits long, and we want to generate $l(> 2n)$ pseudorandom bits. The first bit of the output is the inner product of x and r, $\langle x, r\rangle$. To generate the second bit, compute $h_n^1(x) \leftarrow h(x\|0^{m-n})$, and output $\langle h_n^1(x), r\rangle$. For the third bit, compute $h_n^2(x) \leftarrow h(h_n^1(x)\|0^{m-n})$, and output $\langle h_n^2(x), r\rangle$. Repeat this process until $(l - n)$ bits are output, and also output r.

The latest PRG of this type that relies on reasonable assumptions (regularity and one-wayness) is due to Haitner et al. [11]. In addition to a regular one-way function, each iteration in their scheme uses a new pairwise-independent function (which is basically the only main difference from our construction), whose descriptions are part of the seed of the PRG. Our construction presented above does not use pairwise-independent functions and is thus more efficient, requiring less computation and a significantly shorter seed. Our scheme's security relies on the standard notions of collision-resistance and regularity of the underlying hash function, where the collision-resistance is required to be *exponential* (such a function is also referred in the literature as an "exponentially hard CRHF"). In particular, any polynomial-time adversary should have less than $2^{-n/2}$ probability of finding collisions, where n is the output size of the hash function. This should not be confused with the famous birthday bound, which roughly says that with $2^{n/2}$ number of random trials one can find collisions (with noticeable probability) in any hash function of output size n. Here, we are talking about the probability of collision and not the number of trials.

To the best of our knowledge, this is the first attempt to combine the above two strengthenings (i.e., regularity and exponential hardness) for improving the efficiency of a function-based PRG. While our assumption of exponential collision-resistance is quite strong, unlike the pseudorandomness of hash functions (which not only do not use secret keys, but are usually keyless) ours is still a very well accepted assumption in the community. Also, given the search for a new hash standard SHA-3 by the NIST [22], it is plausible that some (if not all) of the candidate submissions to the competition provide exponential collision-resistance. We later show how to relax the regularity assumption by introducing a new notion that we call *worst-case regularity*. The notion of worst-case regularity lower bounds the size of the smallest set of preimages of different elements in the range, while the common regularity assumption requires all such sets to be of equal size. It was shown by Bellare and Kohno [1] that collision-resistance degrades exponentially (in the range of the function) when a function deviates from regularity, so a CRHF must be very "close" to regular, and experiments

on practical hashes like SHA-1 support this claim (cf. Section 11 in [1]). So, the worst-case regularity assumption on a practical CRHF seems to be reasonable. We note that a notion similar to ours, called "weakly regular" was introduced in [9]. This notion doesn't seem to be useful for our proof, because at a high level it captures the average of the sizes of different preimage sets of a function, while we need a lower bound on these sizes.

Levin [19] observed that the BMY-type constructions are secure for functions that are one-way even when applied on their own outputs, a property called *one-way on iterates* (OWI), which one-way permutations trivially satisfy. However, it would be a stretch to assume that practical hashes have this property. We also note that collision-resistance alone may not be sufficient to prove that a function has the OWI property. Consider a CRHF h that acts as a permutation after one application, i.e. for any x in the domain of h, $h(h(x))$ is a permutation on $h(x)$ (some padding can be used to make $h(x)$ of input size, we omit this padding here for simplicity). For such a CRHF, a security reduction from OWI to collision-resistance is not possible. The reason is that the output of an adversary that can break the OWI security $\left(y \in h^{-1}(h(h(x)))\right)$ cannot be used to find collisions in h, because the set $h^{-1}(h(h(x)))$ has just one element due to h being a permutation after one application. Someone familiar with the proofs of BMY and related PRG constructions may also be skeptical about the other direction, i.e. proving the security of our scheme assuming only the regularity and collision-resistance of h, without employing the "re-randomizing" pairwise-independent functions. The reason is that the security requires h to remain one-way on every iteration, but while h is believed to be collision-resistant and thus one-way (i.e., it is hard to invert $h(x)$ for a random point x in the domain), it is not necessarily hard to invert $h(h(x))$, because $h(x)$ (for a random x) is not necessarily a random point in the domain. In other words, the sets of points to which h is applied may shrink with each iteration, diminishing the one-wayness property of h, and thus violating the security of the PRG. Somewhat surprisingly, we show that these sets in our construction do not shrink significantly, if it is exponentially hard to find collisions in h. Unlike previous results on the security of PRGs, our theorem provides a concrete security statement, so that it is possible to see exactly how the security of our PRG degrades with the degradation in the collision-resistance of the underlying hash function, and thus allows a more accurate comparison with other schemes.

Our construction is very efficient (though still not comparable to practical standardized PRGs [6]) and simple, as at each iteration it uses a hash function and an inner-product computation, both of which are relatively fast. In the full version of this paper [4], we show how using a classical method of [9,7] the efficiency of our scheme can be further improved by extracting up to a constant fraction of n hardcore bits at each iteration, as the underlying CRHF is assumed to be exponentially hard. While our construction is mainly of theoretical interest, we believe our approach and treatment has moved theoretically sound PRGs much further towards practical use. The novel worst-case regularity definition may be of independent interest.

2 Preliminaries

Notation. If f is a function, then $\mathsf{Im}(f)$ denotes the image set of f, and for any $y \in \mathsf{Im}(f)$, $\mathsf{Preim}(f, y)$ denotes the set of preimages of y under f. Let $a, b \in \mathbb{N}$, for simplicity and correctness, we define $\binom{a}{b}$ to be 1 if $a < b$.

2.1 Hash Functions and Their Security

Hash Function. Because of the known difficulties of defining collision-resistance (cf. Section 6.1 in [2]), we follow the standard approach and define hash function families. A *hash function family* H is a collection of functions, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$, such that $m > n$.

Collision-Resistance and Target Collision-Resistance. Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. The *collision-resistance* advantage of an adversary $\mathcal{C}$ attacking H, $\mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{C})$ is defined as

$$\Pr\left[\, h \xleftarrow{\$} H,\ x, x' \xleftarrow{\$} \mathcal{C}(h) :\ x \neq x' \in \{0,1\}^m \bigwedge h(x) = h(x') \,\right].$$

Also, the *target* collision-resistance advantage of an adversary $\mathcal{C}$ attacking H, $\mathbf{Adv}^{\mathrm{tcr}}_H(\mathcal{C})$ is defined as

$$\Pr\left[\, h \xleftarrow{\$} H, x \xleftarrow{\$} \{0,1\}^m, x' \xleftarrow{\$} \mathcal{C}(h, x) :\ x' \in \{0,1\}^m \bigwedge x \neq x' \bigwedge h(x) = h(x') \,\right].$$

Birthday Attack. The birthday attack on a function $f : \{0,1\}^m \to \{0,1\}^n$ is defined in Fig. 1. In this attack, $q \in \mathbb{N}$ points, $x_1, ..., x_q$ are picked independently at random from the domain. If any two of these points form a collision for f, then the attack is successful and those two points are returned. We denote the probability of success of the birthday attack on f by *collision probability*, $\mathbf{CP}(f, q)$. We will slightly abuse the notation sometimes, and use it for function families, where in $\mathbf{CP}(F, q)$ for a function family F, would mean the collision probability of a function picked at random from F.

For $i = 1, ..., q$
  $x_i \xleftarrow{\$} \{0,1\}^m$
  $y_i \leftarrow f(x_i)$
  If $(\exists j :\ j < i \ \bigwedge\ y_i = y_j \ \bigwedge\ x_i \neq x_j)$, return (x_i, x_j).

Fig. 1. Birthday attack (with q trials) on a function $f : \{0,1\}^m \to \{0,1\}^n$

Regularity. A function $f : \{0,1\}^m \to \{0,1\}^n$ is said to be *regular*, if every point in the image set of f have equal number of preimages. Bellare and Kohno introduced the notion of a balance measure, denoted $\mu(f)$ (cf. Section 1 in [1]) to measure the regularity of a function: $\mu(f) = 1$ indicates that the function is fully

regular and $\mu(f) = 0$ means fully irregular (an image point has the maximum number of preimages). The collision probability in the birthday attack for q trials, $\mathbf{CP}(f, q) = \binom{q}{2} \cdot 2^{-n\mu(f)}$ (up to constant factors), so the collision-resistance of any function degrades exponentially (in the range of the function) with the decline in its balance. A CRHF must therefore have a balance close to 1, and experiments on practical hashes like SHA-1 support this claim (cf. Equation 2, Section 11 in [1]). So, SHA-1 and other hash functions (SHA-256, SHA-512, etc.) can be assumed to be *close* to regular. We introduce a notion that we call *worst-case regularity* in Sect. 6 that also captures this closeness.

One-Wayness. Let F be a family of functions, where each $f \in F$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. The *one-way* advantage of an adversary $\mathcal{I}$ attacking F, $\mathbf{Adv}^{\mathrm{ow}}_F(\mathcal{I})$ is defined as

$$\Pr\left[f \xleftarrow{\$} F, x \xleftarrow{\$} \{0,1\}^m, x' \xleftarrow{\$} \mathcal{I}(f, f(x)) : \ x' \in \{0,1\}^m \bigwedge f(x') = f(x) \right].$$

The one-way advantage of a function f (instead of a function family) can be defined similarly: the adversary is given $f(x)$ for a random x, and it has to return an element $x' \in \{0,1\}^m$ such that $f(x') = f(x)$.

Target Collision-Resistance and One-Wayness. The following relation between the notions is well-known.

Theorem 1 ([2], Corollary 5.5). *Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. Then for an adversary $\mathcal{I}$ with running time $t_{\mathcal{I}}$, there exists an adversary $\mathcal{C}$ with running time $t_{\mathcal{C}}$, so that*

$$\mathbf{Adv}^{\mathrm{ow}}_H(\mathcal{I}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{tcr}}_H(\mathcal{C}) + 2^{n-m}, \ \textit{and } t_{\mathcal{C}} \approx t_{\mathcal{I}}.$$

We now present a more general definition that also captures the one-wayness.

Hard to Compute. Let f and g be functions with the same domain $S_m \subseteq \{0,1\}^m$. The *hard-to-compute* advantage of an adversary $\mathcal{I}$ attacking (f, g), $\mathbf{Adv}^{\mathrm{htc}}_{f,g}(\mathcal{I})$ is defined as

$$\Pr\left[x \xleftarrow{\$} S_m : \ \mathcal{I}(f(x)) \in \mathsf{Preim}(g, f(x)) \right].$$

Note that for any adversary $\mathcal{I}$ and any function f, $\mathbf{Adv}^{\mathrm{ow}}_f(\mathcal{I}) = \mathbf{Adv}^{\mathrm{htc}}_{f,\,f}(\mathcal{I})$.

2.2 Hardcore Predicate

Informally, a hardcore predicate of a function is at least as hard to predict as inverting the function itself. Formally, let $g : \{0,1\}^m \to \{0,1\}^n$, $b : \{0,1\}^m \to \{0,1\}$ be two functions, and $a \xleftarrow{\$} \{0,1\}$ be a random bit. The *hardcore predicate* advantage of adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{hcp}}_{g,b}(\mathcal{A})$ is defined as

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^m : \mathcal{A}(g(x), b(x)) = 1 \right] - \Pr\left[x \xleftarrow{\$} \{0,1\}^m : \mathcal{A}(g(x), a) = 1 \right].$$

Here $b(x)$ is called the *hardcore predicate (or bit)* of $g(x)$. In this paper, we use the general hardcore predicate construction of Goldreich and Levin [10], called the "GL-hardcore bit". For two bitstrings x $(= x_1 \| \ldots \| x_m)$ and r $(= r_1 \| \ldots \| r_m)$, define $b(x,r) = \langle x, r \rangle$, the inner product of x and r modulo 2, i.e. $\sum_{i=1}^{m} x_i \cdot r_i \pmod 2$. The following theorem is from [11], and states (using our notation) the security of the GL-hardcore bit.

Theorem 2 (Theorem 2.7, [11]). *Let f and g be functions with the same domain $S_m \subseteq \{0,1\}^m$. For a random $x \in S_m$ and a random $r \in \{0,1\}^m$, define $\widehat{f}$ as $\widehat{f}(x,r) = (f(x), r)$, and its GL-hardcore bit b as $\langle z, r \rangle$, where $z \in \mathsf{Preim}(g, f(x))$ is one of the preimages of $f(x)$ under g. Then for an adversary $\mathcal{A}$ with running time $t_{\mathcal{A}}$, there exists an adversary $\mathcal{I}$ with running time $t_{\mathcal{I}}$, so that*

$$\mathbf{Adv}^{\mathrm{hcp}}_{\widehat{f},b}(\mathcal{A}) \leq 4 \cdot \mathbf{Adv}^{\mathrm{htc}}_{f,g}(\mathcal{I}), \quad \textit{and } t_{\mathcal{I}} = \mathcal{O}\left(m^3 \cdot t_{\mathcal{A}} \cdot \left(\mathbf{Adv}^{\mathrm{hcp}}_{\widehat{f},b}(\mathcal{A})\right)^{-4}\right).$$

2.3 Pseudorandom Generator

Informally, a pseudorandom generator (PRG) is a function that expands a random seed into a longer pseudorandom bit sequence. PRGs were first proposed and constructed by Blum and Micali [3], and Yao [23]. Let $G : \{0,1\}^m \to \{0,1\}^l$ be a function, so that $l > m$. The *prg* advantage of an adversary $\mathcal{P}$ attacking G, $\mathbf{Adv}^{\mathrm{prg}}_{G}(\mathcal{P})$ is defined as

$$\Pr\left[\, s \xleftarrow{\$} \{0,1\}^m : \ \mathcal{P}(G(s)) = 1 \,\right] - \Pr\left[\, y \xleftarrow{\$} \{0,1\}^l : \ \mathcal{P}(y) = 1 \,\right].$$

Here m is the seed length, and l is the number of pseudorandom bits generated.

3 PRG from Iterates

Most of the pseudorandom generators (PRGs) that we know today employ a general design technique: take a function that remains one-way on iterates, and iterate that function for a desired number of times, extracting hardcore bits at every iteration. Below we give a general theorem for the security of such PRGs. The theorem already exists in some form in the cryptographic literature (or, is implied from results in several papers, [19,9,11], to name a few), but we restate it and sketch its proof in the full version of this paper [4] for two main reasons. One is that the proof has evolved over time, starting from Levin's work [19], followed by a proof sketch by Goldreich et al. (cf. Appendix B in [9]), and the improved construction of hard-core predicate by Goldreich and Levin [10]. The second reason is that none of the prior works state the result in its entirety with a concrete security statement.

We will start with a more general definition that also captures the definition of pseudorandomness presented in Sect. 2.3. Let X and Y be random variables

with equal output lengths. Let $\mathcal{D}$ be an adversary for distinguishing X from Y. The *indistinguishability* advantage of $\mathcal{D}$, $\mathbf{Adv}^{\text{ind}}_{X,Y}(\mathcal{D})$ is defined as

$$\mathbf{Adv}^{\text{ind}}_{X,Y}(\mathcal{D}) = \Pr\left[\, x \xleftarrow{\$} X :\ \mathcal{D}(x) = 1 \,\right] \ -\ \Pr\left[\, y \xleftarrow{\$} Y :\ \mathcal{D}(y) = 1 \,\right] .$$

For any adversary $\mathcal{P}$ and any pseudorandom generator G, $\mathbf{Adv}^{\text{ind}}_{G,U_{|G|}}(\mathcal{P}) = \mathbf{Adv}^{\text{prg}}_{G}(\mathcal{P})$, where $U_{|G|}$ is a uniform distribution of size equal to the output size of G.

Theorem 3. *Let $f : \{0,1\}^m \to \{0,1\}^n$ be any function, and for any $i \in \mathbb{N}$, let f^i denote its i^{th} iterate, defined arbitrarily but satisfying the following condition: given only $f^i(x)$ for any $x \in \{0,1\}^m$, $f^{i+1}(x)$ should be efficiently computable. For any $k \in \mathbb{N}$, if f^k is one-way on iterates*[1]*, then for random $x, r \in \{0,1\}^m$, the random variables*

$$X = \langle x, r\rangle \,\|\, \langle f^1(x), r\rangle \,\|\, \ldots \,\|\, \langle f^{k-1}(x),\ r\rangle \,\|r\|\, f^k(x) \quad \textit{and} \quad Y = U_k \|r\| f^k(x)$$

are indistinguishable, where U_k is a uniform distribution of k bits. More formally, for an adversary $\mathcal{D}$ with running time $t_{\mathcal{D}}$, there exists an adversary $\mathcal{I}$ with running time $t_{\mathcal{I}}$, so that

$$\mathbf{Adv}^{\text{ind}}_{X,Y}(\mathcal{D}) \le 8k \cdot \max_{i=1}^{k}\left(\mathbf{Adv}^{\text{htc}}_{f^i,f}(\mathcal{I})\right), \ \textit{and}\ t_{\mathcal{I}} = \mathcal{O}\left(m^3 \cdot t_{\mathcal{D}} \cdot \left(\mathbf{Adv}^{\text{ind}}_{X,Y}(\mathcal{D})\right)^{-4}\right).$$

Informally, the above theorem states that $\langle x, r\rangle \,\|\, \langle f^1(x), r\rangle \,\|\, \ldots \,\|\, \langle f^{k-1}(x),\ r\rangle$ is pseudorandom, given r and $f^k(x)$.

4 Our PRG Construction

We first define the *subset iterate*, a particular way to iterate a hash function on a subset of the actual domain. We use this in our PRG construction.

Subset Iterate. Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. For any $i \in \mathbb{N}$ and any $h \in H$, we define the i^{th} *subset iterate* of h, h^i_n, and denote the corresponding family by H^i_n. For $x \in \{0,1\}^n$, h^i_n is defined recursively as

$$h^1_n(x) = h(x\|0^{m-n}) ,$$
$$h^i_n(x) = h\left(h^{i-1}_n(x)\|0^{m-n}\right) \quad \forall i > 1 .$$

Any unambiguous padding (in place of zeroes, above) can be used to make the input to h of size m bits. For any $i \in \mathbb{N}$, we define the *one-way on iterates or owi* advantage of an adversary $\mathcal{I}$ attacking H^i_n, $\mathbf{Adv}^{\text{owi}}_{H^i_n}(\mathcal{I})$ as

$$\Pr\left[\, h \xleftarrow{\$} H, x \xleftarrow{\$} \{0,1\}^n, x' \xleftarrow{\$} \mathcal{I}\left(h, h^i_n(x)\right) :\ h(x'\|0^{m-n}) = h^i_n(x) \,\right].$$

We now present our PRG construction.

[1] f^k is one-way on iterates, if given $f^k(x)$ for a random $x \in \{0,1\}^m$, it is hard to compute $x' \in \{0,1\}^m$ such that $f(x') = f^k(x)$.

Construction 4. *Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. For any $l > 2n$, a random $h \in H$, which we assume becomes publicly known, and a random seed $s \in \{0,1\}^{2n}$, the pseudorandom generator G parses the input s as $x\|r$, such that both x and r are n-bit strings, and outputs*

$$\langle x, r\rangle \,\|\, \langle h_n^1(x), r\rangle \,\|\, \ldots \,\|\, \langle h_n^{l-n-1}(x), r\rangle \,\|r\ ,$$

where for two bitstrings x $(= x_1\| \ldots \|x_n)$ and r $(= r_1\| \ldots \|r_n)$, $\langle x, r\rangle = \sum_{i=1}^{n} x_i \cdot r_i \ (mod\ 2)$ is their inner product modulo 2.

Note that the seed length of G is $2n$, and it is independent of the output length l. We now present the security analysis of the above construction. For simplicity, in the following theorem we assume that the underlying hash function family is regular. We will show how to relax this assumption to worst-case regularity in Sect. 6.

Theorem 5. *Let H be a hash function family, where each $h \in H$ is a regular function from $\{0,1\}^m$ to $\{0,1\}^n$ and takes time t_H in computation. For any $l > 2n$, let G be the associated PRG, as defined by Construction 4. Then for an adversary $\mathcal{P}$ with running time $t_{\mathcal{P}}$, there exists an adversary $\mathcal{C}$ with running time $t_{\mathcal{C}}$, and $q = \lfloor t_{\mathcal{C}}/t_H \rfloor$, so that*

$$\mathbf{Adv}_G^{\mathrm{prg}}(\mathcal{P}) \le 24 \cdot (l-n) \cdot \left[\binom{\lfloor q/(l-n) - 2 \rfloor}{2}^{-1} \cdot 2^n \cdot \left(\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C})\right)^2 \right]^{\frac{1}{3}} ,$$

$$\text{and } t_{\mathcal{C}} = \max\left\{ \mathcal{O}\left(n^3 \cdot t_{\mathcal{P}} \cdot \left(\mathbf{Adv}_G^{\mathrm{prg}}(\mathcal{P})\right)^{-4} \right), 2(l-n)t_H \right\}.$$

Remark. The above advantage equation is meaningful only if $\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C}) < 2^{-n/2}$. Also, as pointed out in the proof of Theorem 6, the above advantage expression can be made tighter (i.e., $\left(\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C})\right)^2$ could be replaced with $\mathbf{Adv}_H^{\mathrm{tcr}}(\mathcal{C}_1) \cdot \mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C}_2)$ for $\mathcal{C}_1, \mathcal{C}_2$ attacking the target collision-resistance and collision-resistance of H, respectively), though the expression would become even more complicated.

5 Proof of Theorem 5

We start with a short overview of the proof. The proof consists of two main parts: first we prove that the subset iterate used in the construction of our PRG is one-way on iterates (Theorem 6), and then we use the general result of Levin [19] (Theorem 3) to show that our PRG is secure.

The subset iterate is constructed using a hash function. Now, suppose that we have an algorithm $\mathcal{I}$ that can invert the subset iterate, i.e. given $(h, h_n^i(x))$ for any $i \ge 2$, random h, and random x, it returns x' such that $h(x'\|0^{m-n}) = h_n^i(x)$. Then, we can use $\mathcal{I}$ to break the target collision-resistance (TCR) of the underlying hash function. The challenge for the TCR attack (h, x) is used to compute $h(x)$, and then $(h, h(x))$ given to $\mathcal{I}$, and assuming that $h(x) \in \mathsf{Im}(h_n^i)$,

with a very high probability the output of $\mathcal{I}$, x' (and x) is a collision instance for h. These steps are similar to those in the proof from [11].

Now, the main challenge is to show that with a non-negligible probability $h(x) \in \mathsf{Im}(h_n^i)$ (Lemma 9). The proof of the above is the crux and the main novelty of our analysis. We basically show that on iteration, the image set of the subset iterate shrinks by only a polynomial fraction, i.e. for any $i \geq 2$, $|\mathsf{Im}(h_n^i)|/|\mathsf{Im}(h_n^{i-1})|$ is a polynomial fraction. For this purpose, we rely on Lemma 7, which says that the collision probability (in the birthday attack) of a subset iterate degrades only by a multiplicative factor of the square of the number of iterations. It may not be obvious, but the size of the image set and the collision probability of any function are closely related, which is precisely the reason why we are able to prove Lemma 7.

In order to prove Theorem 5, we state the following theorem about the OWI security of the subset iterate used in the construction of our PRG. This theorem together with Theorem 3 (by substituting $(l-n)$ for k) will imply Theorem 5. (One might notice some inconsistencies between Theorem 6 and Theorem 3 in the sense that the underlying primitive in the former is a function family, while it is only a function in the latter. We note, however, that Theorem 3 is applicable without any change in the security reduction to our PRG construction from a hash function family.)

Theorem 6. *Let H be a hash function family, where each $h \in H$ is a regular function from $\{0,1\}^m$ to $\{0,1\}^n$ and takes time t_H in computation. For any $i \in \mathbb{N}$, let H_n^i be the associated i^{th} subset iterate function family of H, as defined in Sect. 4. Then for an adversary $\mathcal{I}$ with running time $t_{\mathcal{I}}$, there exists an adversary $\mathcal{C}$ with running time $t_{\mathcal{C}}$, and $q = \lfloor t_{\mathcal{C}}/t_H \rfloor$, so that*

$$\mathbf{Adv}^{\mathrm{owi}}_{H_n^i}(\mathcal{I}) \leq 3 \cdot \left[\binom{\lfloor q/i - 2 \rfloor}{2}^{-1} \cdot 2^n \cdot \left(\mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{C})\right)^2 \right]^{\frac{1}{3}}, \text{ and } t_{\mathcal{C}} = \max\{t_{\mathcal{I}}, 2it_H\}.$$

Proof. We construct an adversary $\mathcal{C}_1$ with running time $t_{\mathcal{C}_1} = t_{\mathcal{I}}$, for attacking the target collision-resistance of H. $\mathcal{C}_1$ is given a random $h \in H$ and a random $x \in \{0,1\}^m$. It runs the adversary $\mathcal{I}$ attacking one-wayness on iterates of H_n^i with input $(h, h(x))$. Let x' be the output of $\mathcal{I}$. If $x \neq x'\|0^{m-n}$ and $h(x) = h(x'\|0^{m-n})$, it returns $x'\|0^{m-n}$.

We state the following three lemmas from which we will derive the inequality of Theorem 6. Lemma 7 gives an upper bound on the collison probability of birthday attack on the subset iterate of a hash function family. Lemma 8 which is similar to Claim 3.3 of [11], states that the set of inputs on which the adversary $\mathcal{I}$ succeeds reasonably well (better than one third of its advantage) is not small (at least two thirds of its advantage) in size. And, Lemma 9 which is similar to Lemma 3.4 of [11], states that the set of inputs that $\mathcal{I}$ should get in the actual experiment ($h_n^i(x)$ for a random $x \in \{0,1\}^n$) and the set of inputs that it actually gets in the above experiment simulated by $\mathcal{C}_1$ ($h(x)$ for a random $x \in \{0,1\}^m$), overlap for the most part.

Lemma 7. *Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$ and takes time t_H in computation. For any $i \in \mathbb{N}$, let H_n^i be the associated i^{th} subset iterate of H, as defined in Sect. 4. Then for any $q \geq 2i$, there exists an adversary $\mathcal{C}_2$ that runs in time (at most) $q \cdot t_H$, such that*

$$\mathbf{CP}(H_n^i, 2) \leq \frac{\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C}_2)}{\binom{\lfloor q/i-2 \rfloor}{2}} .$$

Proof. We know that for any function f with output size n bits and balance measure $\mu(f)$, (upto constant factors) the collision probability for any $t \in \mathbb{N}$ trials, $\mathbf{CP}(f,t) = \binom{t}{2} \cdot 2^{-n\mu(f)}$, see [1] for details. Let $q' = \lfloor q/i - 2 \rfloor$, then

$$\mathbf{CP}(H_n^i, 2) = \frac{\mathbf{CP}(H_n^i, q')}{\binom{q'}{2}} .$$

Also, it is immediate that there exists an adversary $\mathcal{C}'$ running in time equivalent to q' computations of $h_n^i \in H_n^i$, such that

$$\mathbf{Adv}_{H_n^i}^{\mathrm{cr}}(\mathcal{C}') \geq \mathbf{CP}(H_n^i, q') .$$

(In the worst case, $\mathcal{C}'$ could simply run the birthday attack with q' trials.)

Now, given $\mathcal{C}'$ we will construct the adversary $\mathcal{C}_2$ (from the lemma) that runs in time at most $q \cdot t_H$, so that

$$\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C}_2) = \mathbf{Adv}_{H_n^i}^{\mathrm{cr}}(\mathcal{C}') .$$

Note that for any $h_n^i \in H_n^i$, and any $x \neq x' \in \{0,1\}^n$, if $h_n^i(x) = h_n^i(x')$, then there exists $j < i$, such that $h_n^j(x) \neq h_n^j(x')$ and $h_n^{j+1}(x) = h_n^{j+1}(x')$. When $\mathcal{C}'$ returns (x, x'), $\mathcal{C}_2$ computes $y \leftarrow h_n^j(x)$, $y' \leftarrow h_n^j(x')$, and returns $(y \| 0^{m-n}, y' \| 0^{m-n})$. Recall that $y \neq y'$ and $h(y \| 0^{m-n}) = h(y' \| 0^{m-n})$, so the advantage of $\mathcal{C}_2$ is the same as that of $\mathcal{C}'$. Assuming that one computation of $h_n^i \in H_n^i$ requires the same time as i computations of $h \in H$, we have that the running time of $\mathcal{C}_2$ is at most $q \cdot t_H$ $(\geq (i \cdot q' + 2i) \cdot t_H)$, because apart from running $\mathcal{C}'$ (which is equivalent to $i \cdot q'$ computations of $h \in H$), $\mathcal{C}_2$ does $2j (< 2i)$ computations of $h \in H$ to compute its own output. Thus, $\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C}_2)$ is equal to

$$\mathbf{Adv}_{H_n^i}^{\mathrm{cr}}(\mathcal{C}') \geq \mathbf{CP}(H_n^i, q') = \mathbf{CP}(H_n^i, 2) \cdot \binom{q'}{2} = \mathbf{CP}(H_n^i, 2) \cdot \binom{\lfloor q/i - 2 \rfloor}{2},$$

from which the lemma follows. □

Lemma 8. *Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. For any $i \in \mathbb{N}$ and any $h \in H$, let h_n^i be the associated i^{th} subset iterate and H_n^i be the corresponding family, as defined in Sect. 4. For any adversary $\mathcal{I}$, consider the following probabilities in an experiment where a random $h \in H$ and a random $x \in \{0,1\}^n$ are picked, and a set $S \subseteq \mathsf{Im}(h_n^i)$ is defined as*

$$S = \left\{ y \in \mathsf{Im}(h_n^i) : \Pr\left[h\left(\mathcal{I}\left(h, y\right)\right) = y \right] > \frac{1}{3} \cdot \mathbf{Adv}_{H_n^i}^{\mathrm{owi}}(\mathcal{I}) \right\}.$$

Then,

$$\Pr\left[\, h_n^i(x) \in S \,\right] \geq \frac{2}{3} \cdot \mathbf{Adv}^{\mathrm{owi}}_{H_n^i}(\mathcal{I}).$$

The proof is in the full version of this paper [4].

Lemma 9. *Let H be a hash function family, where each $h \in H$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$. For any $i \in \mathbb{N}$ and any $h \in H$, let h_n^i be the associated i^{th} subset iterate and H_n^i be the corresponding family, as defined in Sect. 4. Consider the following probabilities in an experiment where a random $h \in H$ and a random $x \in \{0,1\}^n$ are picked. If for any $T \subseteq \mathsf{Im}(h_n^i)$ and any $\delta \in [0,1]$,*

$$\Pr\left[\, h_n^i(x) \in T \,\right] \geq \delta,$$

then

$$\Pr\left[\, h(x) \in T \,\right] \geq \frac{\delta^2}{2^{n+1} \cdot \mathbf{CP}(h_n^i, 2)}.$$

Proof. We will first compute a lower bound on the collision probability of h_n^i for two trials, $\mathbf{CP}(h_n^i, 2)$. Pick two elements x_1, x_2 uniformly at random from $\{0,1\}^n$, and then compute the probability that both $h_n^i(x_1), h_n^i(x_2)$ are equal and belong to the set T. This probability is clearly a lower bound on $\mathbf{CP}(h_n^i, 2)$, because T is a subset of $\mathsf{Im}(h_n^i)$. The probability that both $h_n^i(x_1), h_n^i(x_2) \in T$ is at least δ^2, and given that $h_n^i(x_1), h_n^i(x_2) \in T$, the probability that $h_n^i(x_1) = h_n^i(x_2)$ is at least $1/|T|$. The reason is that even though x_1, x_2 are uniformly random elements in $\{0,1\}^n$, $h_n^i(x_1), h_n^i(x_2)$ may not[2] be uniformly random elements in T. So, the probability that $h_n^i(x_1) = h_n^i(x_2)$ can be lower bounded by computing the probability of getting the same element, when two elements are picked (with replacement) uniformly at random from the set T. By simple probability theory, the probability of such an event is $1/|T|$. It may however be noted that in the above calculation, we are also counting trivial collisions, i.e. when $x_1 = x_2$. To compensate for this, we subtract 2^{-n} from the above probability. Hence,

$$\mathbf{CP}(h_n^i, 2) \geq \frac{\delta^2}{|T|} - \frac{1}{2^n}. \tag{1}$$

From (1), we have

$$|T| \geq \frac{\delta^2}{\mathbf{CP}(h_n^i, 2) + 2^{-n}} \geq \frac{\delta^2}{2 \cdot \mathbf{CP}(h_n^i, 2)},$$

because $\mathbf{CP}(h_n^i, 2) \geq 2^{-n}$.

For any $h \in H$, $\mathsf{Im}(h_n^i) \subseteq \mathsf{Im}(h)$, and since $T \subseteq \mathsf{Im}(h_n^i)$, we have that $T \subseteq \mathsf{Im}(h)$. Also, since h is a regular function[3] and $\mathsf{Im}(h) \leq 2^n$, we have that

$$\Pr\left[\, h \xleftarrow{\$} H,\ x \xleftarrow{\$} \{0,1\}^m :\ h(x) \in T \,\right] = \frac{|T|}{|\mathsf{Im}(h)|} \geq \frac{|T|}{2^n}. \tag{2}$$

Thus, the statement of the lemma follows. □

[2] These elements are uniformly distributed, only if h_n^i is a regular function.

[3] We note that this is the only point in the proof that relies on the assumption that h is a regular function.

Implication of Lemma 7, Lemma 8, and Lemma 9. Substituting S for T and $\frac{2}{3} \cdot \mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})$ (from Lemma 8) for δ in Lemma 9, we get that for a random $h \in H$, adversary $\mathcal{I}$ and subset S as defined in Lemma 8

$$\Pr\left[h \xleftarrow{\$} H,\ x \xleftarrow{\$} \{0,1\}^m :\ h(x) \in S \right] \geq \frac{\left(\frac{2}{3} \cdot \mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^2}{2^{n+1} \cdot \mathbf{CP}(h^i_n, 2)} \geq \frac{2^2}{3^2} \cdot \frac{\left(\mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^2}{2^{n+1} \cdot \mathbf{CP}(h^i_n, 2)}.$$

The above equation is a lower bound on the probability that for a random $h \in H$ and a random $x \in \{0,1\}^m$, $\mathcal{I}$'s challenge, $h(x)$ belongs to the subset S. From the description of $\mathcal{C}_1$, it is clear that $\mathbf{Adv}^{\mathrm{tcr}}_H(\mathcal{C}_1)$

$$\begin{aligned}
&= \Pr\left[h \xleftarrow{\$} H, x \xleftarrow{\$} \{0,1\}^m, x' \xleftarrow{\$} \mathcal{I}(h, h(x)) : x \neq x' \| 0^{m-n} \bigwedge h(x' \| 0^{m-n}) = h(x) \right] \\
&= \Pr\left[h \xleftarrow{\$} H, x \xleftarrow{\$} \{0,1\}^m, x' \xleftarrow{\$} \mathcal{I}(h, h(x)) : x \neq x' \| 0^{m-n} \mid h(x' \| 0^{m-n}) = h(x) \right] \\
&\quad \times \Pr\left[h \xleftarrow{\$} H, x \xleftarrow{\$} \{0,1\}^m, x' \xleftarrow{\$} \mathcal{I}(h, h(x)) : h(x' \| 0^{m-n}) = h(x) \right].
\end{aligned}$$

Let us denote the two probabilities in the last equation by P_1 and P_2, respectively. So, $\mathbf{Adv}^{\mathrm{tcr}}_H(\mathcal{C}_1) = P_1 \cdot P_2$. We know that

$$\begin{aligned}
P_1 &\geq \Pr\left[z \xleftarrow{\$} \{0,1\}^{m-n} :\ z \neq 0^{m-n} \right] \\
&\geq \frac{2^{m-n} - 1}{2^{m-n}} \geq \frac{1}{2},
\end{aligned}$$

because x is a uniformly random m-bit string, so the probability that the last $m - n$ bits of x are all 0's is at most 2^{n-m}. Also, from Lemma 8, we have that for a random $h \in H$, adversary $\mathcal{I}$ and subset S as defined in Lemma 8

$$\begin{aligned}
P_2 &\geq \Pr\left[h \xleftarrow{\$} H,\ x \xleftarrow{\$} \{0,1\}^m :\ h(x) \in S \right] \cdot \frac{1}{3} \cdot \mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I}) \\
&\geq \frac{2^2}{3^2} \cdot \frac{\left(\mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^2}{2^{n+1} \cdot \mathbf{CP}(h^i_n, 2)} \cdot \frac{1}{3} \cdot \mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I}) \\
&\geq \frac{2^2}{3^3} \cdot \frac{\left(\mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^3}{2^{n+1} \cdot \mathbf{CP}(h^i_n, 2)}.
\end{aligned}$$

The second inequality is from the lower bound on the probability that $\mathcal{I}$'s challenge $h(x)$ belongs to the subset S, as computed above. Thus,

$$\mathbf{Adv}^{\mathrm{tcr}}_H(\mathcal{C}_1) \geq \frac{\left(\mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^3}{3^3 \cdot 2^n \cdot \mathbf{CP}(h^i_n, 2)}.$$

Combining the above inequality with Lemma 7, we have that for any $q \geq 2i$, there exists an adversary $\mathcal{C}_2$ that runs in time (at most) $q \cdot t_H$, such that

$$\mathbf{Adv}^{\mathrm{tcr}}_H(\mathcal{C}_1) \cdot \mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{C}_2) \geq \frac{\binom{\lfloor q/i-2 \rfloor}{2}}{3^3 \cdot 2^n} \cdot \left(\mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^3 .$$

Recall that the running time of $\mathcal{C}_1$, $t_{\mathcal{C}_1} = t_{\mathcal{I}}$. Let $q = \max\{\lfloor t_{\mathcal{I}}/t_H \rfloor, 2i\}$, and let $\mathcal{C}$ denote the adversary (among $\mathcal{C}_1, \mathcal{C}_2$) with higher collision-resistance advantage, i.e. $\mathcal{C} = \mathcal{C}_1$ if $\mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{C}_1) \geq \mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{C}_2)$, otherwise $\mathcal{C} = \mathcal{C}_2$. (Note that we are getting rid of *target* collision-resistance advantage for a simpler theorem statement, albeit at a loss in the security guarantee) Then,

$$\left(\mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{C})\right)^2 \geq \frac{\binom{\lfloor q/i-2 \rfloor}{2}}{3^3 \cdot 2^n} \cdot \left(\mathbf{Adv}^{\mathrm{owi}}_{H^i_n}(\mathcal{I})\right)^3 .$$

The running time of $\mathcal{C}$, $t_{\mathcal{C}} = \max\{t_{\mathcal{I}}, 2it_H\}$, and hence, Theorem 6 follows. □

6 Relaxing the Regularity Assumption

We introduce a new notion that we call worst-case regularity. It captures the lower bound on the size of the smallest set of preimages of elements from the range of a function. The notion appears somewhat similar to the notions of "weakly regular" introduced by Goldreich et al. [9] and "balance measure" introduced by Bellare and Kohno [1]. However, the reason for introducing a new notion (instead of working with the previous ones), is that it seems unlikely that one can find a tight relation between worst-case regularity and balance measure (or, weak regularity), and thus a tight bound for our theorem, for any general function (or, a CRHF in particular). The intuition behind this is that while worst-case regularity measures the lower bound on the size of preimages, the other two notions are related to the average of these sizes. We will first present the formal definition of worst-case regularity, and then adjust the statement of our main theorem for the case when the underlying CRHF is not necessarily regular.

Worst-Case Regularity. Let F be a family of functions, where each $f \in F$ is a mapping from $\{0,1\}^m$ to $\{0,1\}^n$, and let $\alpha \in (0,1]$. We say that F is α-worst-case regular, if for all $f \in F$ and all $y \in \mathsf{Im}(f)$

$$|\mathsf{Preim}(f,y)| \geq \alpha \cdot 2^{m-n} .$$

For a completely regular function family, $\alpha = 1$.

As pointed out before, the only place where the regularity assumption is required for our proof is in (2) of Lemma 9. So, we will first modify this equation and give justification for this modification, and then adjust our main theorem accordingly. For a not-necessarily regular function family (2) changes as follows.

For any $h \in H$ and any $T \subseteq \mathsf{Im}(h)$, if H is α-worst-case regular, then

$$\Pr\left[\, h \xleftarrow{\$} H,\ x \xleftarrow{\$} \{0,1\}^m :\ h(x) \in T \,\right] \geq \frac{\alpha \cdot |T|}{2^n} , \qquad (3)$$

where H is a hash function family as defined in Lemma 9. Since H is α-worst-case regular, the lower bound on the total size of the preimages of elements in T is $(\alpha \cdot 2^{m-n} \cdot |T|)$. So, when an element is picked uniformly at random from a set of size 2^m, the probability that it hits a subset of size $(\alpha \cdot 2^{m-n} \cdot |T|)$ is $\frac{\alpha \cdot |T|}{2^n}$.

Taking the above equation into account, we present the modified main theorem.

Theorem 10 (Modified Theorem 5). *Let H be an α-worst-case regular hash function family, where each $h \in H$ is a function from $\{0,1\}^m$ to $\{0,1\}^n$ and takes time t_H in computation. For any $l > 2n$, let G be the associated pseudorandom generator, as defined by Construction 4. Then for an adversary $\mathcal{P}$ with running time $t_{\mathcal{P}}$, there exists an adversary $\mathcal{C}$ with running time $t_{\mathcal{C}}$, and $q = \lfloor t_{\mathcal{C}}/t_H \rfloor$, so that*

$$\mathbf{Adv}_G^{\mathrm{prg}}(\mathcal{P}) \leq 24 \cdot (l-n) \cdot \left[\binom{\lfloor q/(l-n) - 2 \rfloor}{2}^{-1} \cdot \alpha^{-1} \cdot 2^n \cdot \left(\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{C})\right)^2 \right]^{\frac{1}{3}},$$

$$\text{and } t_{\mathcal{C}} = \max\left\{ \mathcal{O}\left(n^3 \cdot t_{\mathcal{P}} \cdot \left(\mathbf{Adv}_G^{\mathrm{prg}}(\mathcal{P})\right)^{-4} \right), 2(l-n)t_H \right\}.$$

7 Conclusion

We propose a hash-function-based construction of a pseudorandom generator. Our scheme is similar to the "randomized iterate" construction of Haitner et al., but eliminates the need for the use of pairwise-independent functions on each iteration of the PRG. As a result, our PRG is significantly more efficient in terms of computation and the seed length. We first prove the security of our scheme assuming the underlying hash function is regular and collision-resistant, where the collision-resistance is required to be exponential. Then we show how to relax the regularity assumption on the hash function by introducing a new notion called worst-case regularity, which lower bounds the size of the smallest preimage set in a function. Unlike the previous similar schemes, our construction is accompanied by a concrete security statement.

References

1. Bellare, M., Kohno, T.: Hash Function Balance and Its Impact on Birthday Attacks. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 401–418. Springer, Heidelberg (2004), http://eprint.iacr.org/2003/065
2. Bellare, M., Rogaway, P.: Hash Functions. Introduction to Modern Cryptography, ch. 5, http://www-cse.ucsd.edu/users/mihir/cse207/w-hash.pdf
3. Blum, M., Micali, S.: How to Generate Cryptographically Strong Sequences of Pseudo Random Bits. In: FOCS 1982, pp. 112–117. IEEE (1982)
4. Boldyreva, A., Kumar, V.: A New Pseudorandom Generator from Collision-Resistant Hash Functions. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 187–202. Springer, Heidelberg (2012), http://eprint.iacr.org

5. Desai, A., Hevia, A., Yin, Y.L.: A Practice-Oriented Treatment of Pseudorandom Number Generators. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 368–383. Springer, Heidelberg (2002)
6. FIPS PUB 186-2, Digital Signature Standard, National Institute of Standards and Technologies (1994)
7. Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press (2001)
8. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM 33(4), 792–807 (1986)
9. Goldreich, O., Krawczyk, H., Luby, M.: On the Existence of Pseudorandom Generators (Extended Abstract). In: FOCS 1988, pp. 12–24. IEEE (1988); Full version in SIAM Journal of Computing, 22(6), 1163–1175 (1993)
10. Goldreich, O., Levin, L.: A Hard-Core Predicate for all One-Way Functions. In: STOC 1989, pp. 25–32. ACM (1989)
11. Haitner, I., Harnik, D., Reingold, O.: On the Power of the Randomized Iterate. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 22–40. Springer, Heidelberg (2006), http://eccc.hpi-web.de/eccc-reports/2005/TR05-135
12. Haitner, I., Harnik, D., Reingold, O.: Efficient Pseudorandom Generators from Exponentially Hard One-Way Functions. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 228–239. Springer, Heidelberg (2006)
13. Haitner, I., Reingold, O., Vadhan, S.: Efficiency improvements in constructing pseudorandom generators from one-way functions. In: STOC 2010, pp. 437–446. ACM (2010)
14. Håstad, J.: Pseudo-Random Generators under Uniform Assumptions. In: STOC 1990, pp. 395–404. ACM (1990)
15. Håstad, J., Impagliazzo, R., Levin, L., Luby, M.: A Pseudorandom Generator from any One-way Function. SIAM Journal of Computing 28(4), 1364–1396 (1999)
16. Holenstein, T.: Pseudorandom Generators from One-Way Functions: A Simple Construction for Any Hardness. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 443–461. Springer, Heidelberg (2006)
17. Impagliazzo, R., Levin, L., Luby, M.: Pseudo-random Generation from one-way functions (Extended Abstracts). In: STOC 1989, pp. 12–24. ACM (1989)
18. Impagliazzo, R., Nisan, N., Wigderson, A.: Pseudorandomness for network algorithms. In: STOC 1994, pp. 356–364. ACM (1994)
19. Levin, L.: One-way functions and pseudorandom generators. Combinatorica 7(4), 357–363 (1987)
20. Naor, M.: Bit Commitment Using Pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991)
21. Nisan, N.: Pseudorandom generators for space-bounded computation. Combinatorica 12(4), 449–461 (1992)
22. SHA-3: Cryptographic Hash Algorithm Competition. National Institute of Standards and Technology (2008), http://csrc.nist.gov/groups/ST/hash/sha-3/index.html
23. Yao, A.: Theory and Applications of Trapdoor Functions (Extended Abstract). In: FOCS 1982, pp. 80–91. IEEE (1982)

PMAC with Parity: Minimizing the Query-Length Influence

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation, Japan
yasuda.kan@lab.ntt.co.jp

Abstract. We present a new variant of PMAC (Parallelizable Message Authentication Code). The new mode calls an n-bit block cipher using four different block-cipher keys but attains a security bound of a novel form $O(q^2/2^n + \ell\sigma q/2^{2n})$. Here, q denotes the total number of queries, ℓ the maximum length of each query (in blocks), and σ the total query complexity (in blocks). Our bound improves over the previous PMAC security $O(\ell q^2/2^n)$ from FSE 2007 and over $O(\sigma q/2^n)$ from FSE 2010. Moreover, when $\ell > 2^{n/6}$, our bound holds valid for larger values of q than the beyond-birthday bound $O(\ell^3 q^3/2^{2n})$ does—the bound of the PMAC variant from CRYPTO 2011. In particular, our bound becomes "ℓ-free" as $O(q^2/2^n)$ under the condition that all queries are shorter than $2^{n/2}$ blocks (*i.e.*, $\ell \leq 2^{n/2}$). Our construction is fairly efficient; it runs at rate 2/3 (meaning 1.5 encryptions to process n bits), which can be made even faster by increasing the number of keys. Thus our construction brings substantial gain in security guarantee without much loss in efficiency, which becomes especially valuable for 64-bit block ciphers.

Keywords: Block cipher, permutation, mode of operation, provable security, game-playing technique, checksum.

1 Introduction

Message Authentication Codes (MACs) are often realized via some modes of operation using n-bit block ciphers, where typically we have $n = 64$ or $n = 128$. Prominent modes are CBC MACs (*e.g.*, [2,15,6,11,9,13]) and PMAC [7,17,18].

Most of these block-cipher MAC constructions are provided with proofs of security, which generally guarantee the level of "birthday security." The basic birthday bounds look like $O(\ell^2 q^2/2^n)$ or $O(\sigma^2/2^n)$,[1] where q is the total number of (chosen-message) queries (to the MAC oracle), ℓ the maximum length of each query, and σ the total length of all queries (The lengths are measured in terms of the number of blocks).

The basic birthday bounds often become insufficient, especially when $n = 64$. In legacy systems or lightweight applications where 64-bit block ciphers are used, it is desirable to provide a higher security guarantee. Roughly speaking, we see

[1] With abuse of notation we use the big-O notation to mean that constant coefficients are omitted.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 203–214, 2012.

that there are two aspects of improving the birthday bounds—improving the ℓ-factor in the bounds, or the q-factor.

Improving in ℓ: Minimizing the Query-Length Influence. The basic birthday bound $O(\ell^2 q^2/2^n)$ would become void when $\ell \approx 2^{n/2}$. This limitation on ℓ can be relaxed in multiple ways.

The first is to provide better security analysis, as done for many of the MAC constructions. Frequently, bounds of the form $O(\sigma^2/2^n)$ can be obtained, rather than $O(\ell^2 q^2/2^n)$. Bounds of the form $O(\ell q^2/2^n)$ are proven for CBC MAC [3] and for PMAC [12]. Even better bounds $O(\sigma q/2^n)$ are achieved for a wide class of block-cipher MAC constructions [14].

The second is to provide constructions that make use of a counter. The idea of using a counter appears in previous constructions such as XOR MAC [1] and PCS [5]. For example, the following PMAC-type construction yields an "ℓ-free" bound $O(q^2/2^n)$: For simplicity assume that n is even; divide a (padded) message M into $n/2$-bit blocks as $M[1], M[2], \ldots, M[m]$; then compute $C[i] \leftarrow E_{K_1}\big(i \| M[i]\big)$, where E_{K_1} is an n-bit block cipher using a key K_1, which encrypts the counter i encoded into an $n/2$-bit string and concatenated to the message block $M[i]$; finally output the tag value $T \leftarrow E_{K_2}\big(C[1] \oplus \cdots \oplus C[m]\big)$, where $\oplus$ means bitwise xor. Unfortunately this construction runs at rate 1/2 (meaning two encryptions to process n bits), and by specification the maximum message length ℓ is limited to $2^{n/2}$ blocks.

The third is to utilize randomization (*e.g.*, [8]), which also yields an ℓ-free $O(q^2/2^n)$ bound. Such a scheme requires a (pseudo-)random number generator and must attach each generated random salt to each tag, which results in a larger tag size.

Improving in q: Maximizing the Query-Number Acceptance. All of the above bounds so far have the limitation $q < 2^{n/2}$. Unlike the case of ℓ, this is inevitable for classical MAC constructions that possess n-bit intermediate state values, because for such iterated MACs there exists a generic attack that can produce a forgery using about $2^{n/2}$ queries [16].

To get rid of the limitation $q < 2^{n/2}$, one must come up with new constructions that achieve so-called beyond-birthday security. Such constructions exist [10,20,21], achieving $O(\ell^3 q^3/2^{2n})$ bounds. So these constructions remain secure up to $O(2^{2n/3})$ query complexity.

Unbalance between q and ℓ. Let us look more closely at the two bounds $O(\ell q^2/2^n)$ and $O(\ell^3 q^3/2^{2n})$. It is not the case that one of them is better than the other for all parameters of ℓ and q. Indeed, we have seen that, when $\ell = 1$, the former provides only the birthday security $O(q^2/2^n)$, whereas the latter $O(q^3/2^{2n})$. However, when $\ell = 2^{2n/3}$, the former still gives us (some) security $O(q^2/2^{n/3})$, whereas the latter beyond-birthday bound vanishes completely.

It depends on each application which factor, ℓ or q, is more important. In the current work we focus on the situations where the ℓ-factor is more important. For example, consider the case of 64-bit block ciphers. The figure $\ell = 2^{n/2} = 2^{32}$

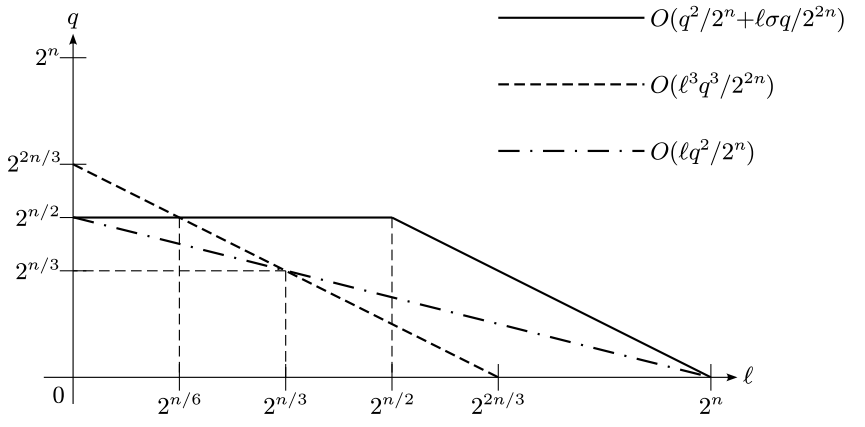


Fig. 1. Values of ℓ and q (unequally scaled) that make the three bounds vacuous

corresponds to 32 GB, whereas $q = 2^{32} \approx 4.3 \times 10^9$ corresponds to about 136 years if executed every second. Then our target is, for example, those systems that handle data of gigabyte sizes but produce at most one tag per second. We are unable to list specific examples of such targets but believe that a number of security applications fall into the category.

Our Contributions. We present a new variant of PMAC which attains a security bound of a novel form $O(q^2/2^n + \ell\sigma q/2^{2n})$. This bound improves over the previous bounds in terms of ℓ-factor. The new construction has the following features:

1. The scheme does not use randomization.
2. The algorithm can handle messages having $\ell > 2^{n/2}$ blocks.
3. The basic version of our construction runs at rate 2/3 (meaning 1.5 encryptions to process n bits).
4. The new bound improves over $O(\ell\sigma/2^n)$ for all values of ℓ and q.
5. The new bound improves over $O(\ell^3q^3/2^{2n})$ for the following values of ℓ and q:
   (a) when $\ell > 2^{n/3}$,
   (b) when $2^{n/3} \geq \ell \geq 2^{n/6}$ and $q > 2^n/\ell^3$.

See Fig. 1, which plots the values of ℓ and q that voids the bounds $O(q^2/2^n + \ell\sigma q/2^{2n})$, $O(\ell^3q^3/2^{2n})$ and $O(\ell q^2/2^n)$. For $n = 64$, $\ell = 2^{n/6}$ corresponds to about 12.7 kB and $\ell = 2^{n/3}$ to 20.2 MB. In practice, one is interested more in the curve for which there is some (*e.g.*, $1/2^{32}$) security left rather than in the curve for which the security vanishes. The diagram indicating such locations would become essentially the same, except that it is drawn on a sliding scale.

Unfortunately, the new construction has some serious disadvantages:

1. The basic version uses four independent block-cipher keys.
2. Different versions run faster at rate 3/4, 4/5, ..., but using five, six, ... different keys.
3. The new construction requires larger memory to store intermediate state values.

These undesirable features may prevent us from applying the new construction to some of the lightweight or legacy systems using 64-bit block ciphers. However, the current work still makes a significant contribution to the construction of block-cipher MACs, demonstrating a new tradeoff between performance and security.

Organization of the Paper. Necessary symbols and security notions are given in Sect. 2. We start with the basic version, a rate-2/3 construction, which is defined in Sect. 3. Its security proof is given in Sect. 4. In Sect. 5 we describe other (rate-3/4, 4/5, etc.) versions. We end the paper by making some remarks in Sect. 6.

2 Preliminaries

Notation System. Let the symbol $\mathrm{Perm}(n)$ denote the set of all permutations $P : \{0,1\}^n \to \{0,1\}^n$. Similarly, let $\mathrm{Func}(n)$ denote the set of all functions $F : \{0,1\}^n \to \{0,1\}^n$. Fix a key space $\mathbb{K}$. Usually $\mathbb{K} = \{0,1\}^k$, where $k = 80, 128$, 192 or 256. We define a blockcipher E as a function $E : \mathbb{K} \times \{0,1\}^n \to \{0,1\}^n$ such that for each key $K \in \mathbb{K}$ the specified function E_K is in $\mathrm{Perm}(n)$. Here $E_K : \{0,1\}^n \to \{0,1\}^n$ is defined as $E_K(X) := E(K,X)$. We write E_K^{-1} for the inverse permutation.

We sometimes treat $\{0,1\}^n$ as a set of integers $\{0,1,\ldots,2^n-1\}$. This can be done by converting an n-bit string $a_{n-1}\cdots a_1 a_0 \in \{0,1\}^n$ to an integer $a_{n-1}2^{n-1} + \cdots + a_1 2 + a_0$, where multiplication and addition are arithmetic (modulo 2^n.)

We let $GF(2^n)$ be the finite field having 2^n elements. We treat $\{0,1\}^n$ also as $GF(2^n)$. That is, we identify an n-bit string $a_{n-1}\cdots a_1 a_0 \in \{0,1\}^n$ with a formal polynomial $a_{n-1}\mathtt{x}^{n-1} + \cdots a_1\mathtt{x} + a_0 \in GF(2)[\mathtt{x}]$. For this we fix an irreducible polynomial $a(\mathtt{x}) = \mathtt{x}^n + a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0 \in GF(2)[\mathtt{x}]$. For example we can choose irreducible polynomials $a(\mathtt{x}) = \mathtt{x}^{64} + \mathtt{x}^4 + \mathtt{x}^3 + \mathtt{x} + 1$ for $n = 64$ and $a(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$ for $n = 128$. These are actually primitive polynomials, meaning the element $2 = \mathtt{x}$ generates the entire multiplicative group $GF(2^n)^*$ of order $2^n - 1$.

Security Definitions. In this paper an adversary $\mathcal{A}$ is an oracle machine. We write $\mathcal{A}^{\mathcal{O}(\cdot)} = y$ to denote the event that $\mathcal{A}$ outputs y after interacting with an oracle $\mathcal{O}(\cdot)$. We measure the resources of $\mathcal{A}$ in terms of time and query complexities. We fix a model of computation and a method of encoding.

The query complexity is measured in terms of the number q of queries, in terms of the maximum length ℓ of each query, and in terms of the total query complexity σ. The resources ℓ and σ are measured in blocks (n bits).

We say that (informally) a block cipher E is a (secure) pseudo-random permutation (PRP) if it is indistinguishable from a random permutation $P \xleftarrow{\$} \mathrm{Perm}(n)$, where $\xleftarrow{\$}$ means uniformly random sampling. Specifically, we consider the advantage function

$$\mathrm{Adv}^{\mathrm{prp}}_{E}(\mathcal{A}) := \Pr\left[\mathcal{A}^{E_K(\cdot)} = 1; K \xleftarrow{\$} \mathbb{K}\right] - \Pr\left[\mathcal{A}^{P(\cdot)} = 1; P \xleftarrow{\$} \mathrm{Perm}(n)\right],$$

and if this quantity is "small enough" for a class of adversaries, then we say that E is a "secure" PRP. Here note that the probabilities are defined over internal coin tosses of $\mathcal{A}$, if any, as well as over the choices of K and P. We further define $\mathrm{Adv}^{\mathrm{prp}}_{E}(t, q) := \max_{\mathcal{A}} \mathrm{Adv}^{\mathrm{prp}}_{E}(\mathcal{A})$, where the max runs over adversaries $\mathcal{A}$ whose time complexity is at most t, making at most q queries to its oracle.

With abuse of notation let $\{0,1\}^*$ denote the set of finite bit strings whose length is at most ℓ blocks. Let $\mathrm{Func}(*, n)$ denote the set of functions $G : \{0,1\}^* \to \{0,1\}^n$. Our goal is to construct a pseudo-random function (PRF) $F_K : \{0,1\}^* \to \{0,1\}^n$ having keys $K \in \mathbb{K}'$. Recall that any PRF can be used as a secure MAC. We say that F is a secure PRF if it is indistinguishable from a random function $G \xleftarrow{\$} \mathrm{Func}(*, n)$, or more precisely, we define

$$\mathrm{Adv}^{\mathrm{prf}}_{F}(\mathcal{A}) := \Pr\left[\mathcal{A}^{F_K(\cdot)} = 1; K \xleftarrow{\$} \mathbb{K}'\right] - \Pr\left[\mathcal{A}^{G(\cdot)} = 1; G \xleftarrow{\$} \mathrm{Func}(*, n)\right].$$

We also define $\mathrm{Adv}^{\mathrm{prf}}_{F}(t, q, \ell, \sigma)$ to be the maximum advantage running over adversaries $\mathcal{A}$ whose resources are limited to t, q, ℓ, σ.

Game-Playing Techniques and Lazy Sampling. Our security proofs are based on the game-playing techniques [4]. We perform lazy sampling for a random permutation $P \xleftarrow{\$} \mathrm{Perm}(n)$. That is, P is initially everywhere undefined, and when a value $P(X)$ becomes necessary at some point in the game, a corresponding range point C is randomly sampled as $C \xleftarrow{\$} \{0,1\}^n$.

3 Description of the New Mode

In this section we define the rate-2/3 version of our PMAC variant. See Algorithm 1 and Fig. 2. The algorithm uses four permutations P_1, P_2, P_3 and P_4, which are in practice realized via a block cipher using four keys.

In pre-computation stage the algorithm `PMAC2/3` prepares mask values as $L_1 \leftarrow P_1(0)$, $L_2 \leftarrow P_2(0)$ and $L_3 \leftarrow P_3(0)$. These values are updated via finite-field multiplication by 2 as $2^i L_1$, $2^i L_2$ and $2^i L_3$.

The algorithm takes a message input M and adds the usual 10^* padding so that the number of (n-bit) blocks becomes an even number. The padded message $M \| 10^*$ is then divided into n-bit blocks as $M[1], \ldots, M[2m]$ before being processed.

Input: a message $M \in \{0,1\}^*$
Output: a tag T
$L_1 \leftarrow P_1(0)$; $L_2 \leftarrow P_2(0)$; $L_3 \leftarrow P_3(0)$
$\big(M[1], \ldots, M[2m]\big) \leftarrow M \| 10^*$
for $i = 1$ **to** m **do**
  $X[2i-1] \leftarrow M[2i-1] \oplus 2^{i-1}L_1$; $X[2i] \leftarrow M[2i] \oplus 2^{i-1}L_2$
  $C[2i-1] \leftarrow P_1\big(X[2i-1]\big)$; $C[2i] \leftarrow P_2\big(X[2i]\big)$
  $Y[i] \leftarrow M[2i-1] \oplus M[2i] \oplus 2^{i-1}L_3$
  $D[i] \leftarrow P_3\big(Y[i]\big)$
end
$S \leftarrow C[1] \oplus \cdots \oplus C[2m] \oplus D[1] \oplus \cdots \oplus D[m]$
$T \leftarrow P_4(S)$
return T

Algorithm 1: The rate-2/3 construction $\mathtt{PMAC2/3}[P_1, P_2, P_3, P_4]$

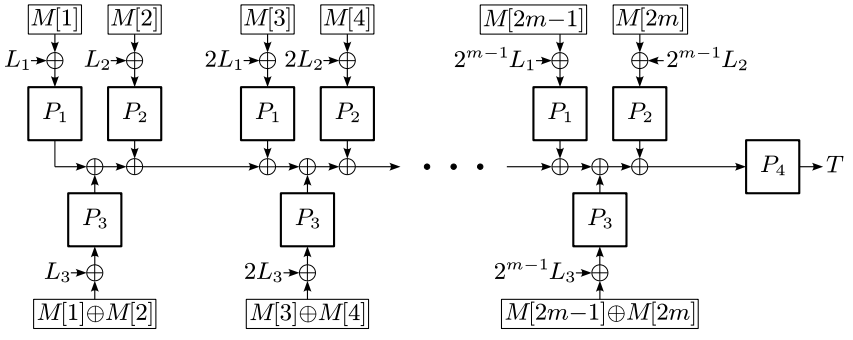


Fig. 2. A pictorial representation of the rate-2/3 construction

The rest of the process is PMAC-like, except that 1) we use different keys between odd-numbered blocks and even-numbered blocks, 2) we have extra "checksum" blocks $M[2i-1] \oplus M[2i]$ for which another key is used, and 3) we use yet another key for the finalization.

The exact amount of memory to run the rate-2/3 construction depends on the specific implementation. Generally, the different keys K_2, K_3, K_4, the different mask values L_2, L_3, and the checksum block $M[2i-1] \oplus M[2i]$ are the state values that need extra memory.

4 Security Proofs

We now prove our security result for the rate-2/3 construction; we prove that the algorithm $\mathtt{PMAC2/3}[E_{K_1}, E_{K_2}, E_{K_3}, E_{K_4}]$ is a secure PRF having a bound of the form $O(q^2/2^n + \ell\sigma q/2^{2n})$:

Theorem 1. *The algorithm* $\mathtt{PMAC2/3}[E_{K_1}, E_{K_2}, E_{K_3}, E_{K_4}]$ *is a secure PRF on the assumption that the underlying block cipher* E *is a secure PRP. More precisely, we have*

initialize: $P_1, P_2, P_3 \xleftarrow{\$} \mathrm{Perm}(n)$, $F \xleftarrow{\$} \mathrm{Func}(n)$
on α-*th query* $M^{(\alpha)}$ **do**
  $S^{(\alpha)} \leftarrow \mathtt{inner}[P_1, P_2, P_3](M^{(\alpha)})$
  $T \leftarrow F(S^{(\alpha)})$
  if $S^{(\beta)} = S^{(\alpha)}$ *for some* $\beta \in \{1, \ldots, \alpha - 1\}$ **then**
    if $\neg$bad **then**
      $\mathtt{coll}(\beta, \alpha) \leftarrow$ **true**
    end
    $\mathtt{bad} \leftarrow$ **true** $\boxed{T \xleftarrow{\$} \{0,1\}^n}$
  end
  return T
end

Algorithm 2: Games G_0 (with the boxed statement) and G_1 (without)

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathtt{PMAC2/3}[E_{K_1}, E_{K_2}, E_{K_3}, E_{K_4}]}(t, q, \ell, \sigma) \leq \frac{q^2}{2^n} + \frac{\ell\sigma q}{2^{2n}} + 4\,\mathrm{Adv}^{\mathrm{prp}}_{E}(t', \sigma),$$

where t' is t plus the time complexity to compute the E algorithm σ times.

Proof. We consider the algorithm $\mathtt{PMAC2/3}[P_1, P_2, P_3, F]$ where P_1, P_2, P_3 are three independent random permutations and F an independent random function.

Let $\mathcal{A}$ be an adversary playing the games defined in Algorithm 2. We limit the resources of $\mathcal{A}$ by t, q, ℓ, σ. In the games we use the algorithm `inner`, which is defined to be the subroutine of `PMAC2/3` that outputs S, so that we have

$$\mathtt{PMAC2/3}[P_1, P_2, P_3, F](M) = F\big(\mathtt{inner}[P_1, P_2, P_3](M)\big).$$

Without loss of generality we assume that adversary $\mathcal{A}$ never repeats its queries.

We observe that on one hand game G_0 corresponds to a truly random function mapping $\{0,1\}^*$ to $\{0,1\}^n$. On the other hand, game G_1 corresponds to the algorithm $\mathtt{PMAC2/3}[P_1, P_2, P_3, F]$. Therefore, by the fundamental lemma of game playing, we get

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{prf}}_{\mathtt{PMAC2/3}[P_1, P_2, P_3, F]}(\mathcal{A}) &= \Pr\Big[G_1(\mathcal{A}) \text{ outputs } 1\Big] - \Pr\Big[G_0(\mathcal{A}) \text{ outputs } 1\Big] \\
&\leq \Pr\Big[G_1(\mathcal{A}) \text{ sets } \mathtt{bad}\Big] \\
&= \Pr\left[\bigvee_{\beta<\alpha} \Big(G_1(\mathcal{A}) \text{ sets } \mathtt{coll}(\beta, \alpha)\Big)\right] \\
&\leq \sum_{\beta<\alpha} \Pr\Big[G_1(\mathcal{A}) \text{ sets } \mathtt{coll}(\beta, \alpha)\Big],
\end{aligned}$$

so in the following we evaluate the probability $\Pr\Big[G_1(\mathcal{A}) \text{ sets } \mathtt{coll}(\beta, \alpha)\Big]$.

initialize: $P_1, P_2, P_3 \xleftarrow{\$} \mathrm{Perm}(n)$
$(M, M') \leftarrow \mathcal{B}$
$S \leftarrow \mathtt{inner}[P_1, P_2, P_3](M)$
$S' \leftarrow \mathtt{inner}[P_1, P_2, P_3](M')$
if $M \neq M'$ *and* $S = S'$ **then**
  $\mathtt{bad} \leftarrow$ **true**
end

Algorithm 3: Game G for adversary $\mathcal{B}$

To do this, let us consider game G defined in Algorithm 3. The goal of the adversary $\mathcal{B}$ playing game G is to find an inner collision $S = S'$ for different messages $M \neq M'$.

Let us construct an adversary $\mathcal{B}_{\beta,\alpha}$ that uses $\mathcal{A}$ and plays game G. The adversary $\mathcal{B}_{\beta,\alpha}$ runs $\mathcal{A}$ and returns random strings to $\mathcal{A}$'s oracle queries. At the β-th query, $\mathcal{B}_{\beta,\alpha}$ stores the query $M^{(\beta)}$ and resumes $\mathcal{A}$. At the α-th query, $\mathcal{B}_{\beta,\alpha}$ stops $\mathcal{A}$ and outputs $(M^{(\beta)}, M^{(\alpha)})$. If $\mathcal{A}$ is to set $\mathtt{coll}(\beta, \alpha)$, then we observe that $\mathcal{B}_{\beta,\alpha}$ correctly simulates game G_1 for $\mathcal{A}$, up to $(\alpha - 1)$-th query, and $\mathcal{B}_{\beta,\alpha}$ must set $\mathtt{bad}$ in game G. Therefore, we get

$$\Pr\Big[G_1(\mathcal{A}) \text{ sets } \mathtt{coll}(\beta, \alpha)\Big] \leq \Pr\Big[G(\mathcal{B}_{\beta,\alpha}) \text{ sets } \mathtt{bad}\Big],$$

and in the following lemma we evaluate the quantity $\Pr\Big[G(\mathcal{B}) \text{ sets } \mathtt{bad}\Big]$:

Lemma 1 (Main Lemma). *For two messages $M, M' \in \{0,1\}^*$ such that $M \neq M'$, we have*

$$\Pr\Big[\mathtt{inner}[P_1, P_2, P_3](M) = \mathtt{inner}[P_1, P_2, P_3](M'); \\ P_1, P_2, P_3 \xleftarrow{\$} \mathrm{Perm}(n)\Big] \leq \frac{1}{2^n} + \frac{m^2}{2^{2n}},$$

where m is half the block length of the longer message (M or M').

Proof. We prove this lemma by lazy sampling of P_1, P_2, P_3. Without loss of generality assume $m \geq m'$.

Case $m > m'$. We focus on the last input blocks $X[2m-1], X[2m]$. We sample $P_1(0)$ and $P_2(0)$. The probability that both of the inputs $X[2m-1], X[2m]$ collide with some other inputs to P_1, P_2 (including the zero input 0) is at most

$$\frac{m}{2^n} \cdot \frac{m}{2^n} = \frac{m^2}{2^{2n}}.$$

Otherwise at least one of the inputs must get sampled in computing the state value S or S'. In that case the probability that $S = S'$ happens is at most $1/2^n$, which proves the lemma for this case.

Case $m = m'$. Let i be the maximum index such that $\big(M[2i-1], M[2i]\big) \neq \big(M'[2i-1], M'[2i]\big)$ (The value i is uniquely determined as soon as the two messages M and M' are fixed.) We also look at the checksum blocks $M[2i-1] \oplus M[2i]$ and $M'[2i-1] \oplus M'[2i]$. We observe that, of these three blocks in comparison, at least two of them are different (between M and M'); so choose two different blocks (according to some fixed order). Sample $P_1(0)$, $P_2(0)$ and $P_3(0)$. The probability that both of that different input blocks collide with somewhere else (including zero) is at most

$$\frac{m}{2^n} \cdot \frac{m}{2^n} = \frac{m^2}{2^{2n}}.$$

Otherwise at least one of the inputs must get sampled in computing the state value S or S'. In that case the probability that $S = S'$ happens is at most $1/2^n$. So this case is also proven. □

Now we go back to computing the overall probability. We would like to evaluate the quantity

$$\sum_{\beta<\alpha} \Pr\Big[G(\mathcal{B}_{\beta,\alpha}) \text{ sets } \mathtt{bad}\Big] \leq \sum_{\beta<\alpha} \left(\frac{1}{2^n} + \frac{m^2}{2^{2n}}\right),$$

where m is half the block size of the longer message, either $M^{(\beta)}$ or $M^{(\alpha)}$. We reorder the messages so that $M^{(1)}$, $M^{(2)}$, …, $M^{(q)}$ are in (weakly) length-increasing order, $M^{(1)}$ being (one of) the shortest message and $M^{(q)}$ (one of) the longest. Now we have

$$\begin{aligned}
\sum_{\beta<\alpha} \left(\frac{1}{2^n} + \frac{{m^{(\alpha)}}^2}{2^{2n}}\right) &= \sum_{\beta<\alpha} \frac{1}{2^n} + \sum_{\beta<\alpha} \frac{{m^{(\alpha)}}^2}{2^{2n}} \\
&\leq \binom{q}{2} \cdot \frac{1}{2^n} + \frac{\ell+1}{2} \cdot \sum_{\beta<\alpha} \frac{m^{(\alpha)}}{2^{2n}} \\
&\leq \frac{q^2}{2^{n+1}} + \frac{\ell+1}{2} \cdot \sum_{\alpha=2}^{q} \sum_{\beta=1}^{\alpha-1} \frac{m^{(\alpha)}}{2^{2n}} \\
&\leq \frac{q^2}{2^{n+1}} + \frac{\ell+1}{2} \cdot \sum_{\alpha=2}^{q} \frac{m^{(\alpha)}(\alpha-1)}{2^{2n}} \\
&\leq \frac{q^2}{2^{n+1}} + \frac{\ell+1}{2} \cdot \frac{(\sigma-1)(q-1)}{2^{2n}} \\
&\leq \frac{q^2}{2^{n+1}} + \frac{\ell\sigma q}{2^{2n}},
\end{aligned}$$

to which we add terms $q^2/2^{n+1}$ (PRP/PRF switching lemma [4] for P_4), and $4\,\mathrm{Adv}^{\mathrm{prp}}_E(t', \sigma)$ (replacing P_1, P_2, P_3, P_4 with $E_{K_1}, E_{K_2}, E_{K_3}, E_{K_4}$). This would give us the desired bound. □

Input: a message $M \in \{0,1\}^*$
Output: a tag T
$L_1 \leftarrow P_1(0)$; $L_2 \leftarrow P_2(0)$; $L_3 \leftarrow P_3(0)$; $L_4 \leftarrow P_4(0)$
$\big(M[1], \ldots, M[3m]\big) \leftarrow M \| 10^*$
for $i = 1$ **to** m **do**
  $X[3i-2] \leftarrow M[3i-2] \oplus 2^{i-1}L_1$; $X[3i-1] \leftarrow M[3i-1] \oplus 2^{i-1}L_2$;
  $X[3i] \leftarrow M[3i] \oplus 2^{i-1}L_3$
  $C[3i-2] \leftarrow P_1\big(X[3i-2]\big)$; $C[3i-1] \leftarrow P_2\big(X[3i-1]\big)$; $C[3i] \leftarrow P_3\big(X[3i]\big)$
  $Y[i] \leftarrow M[3i-2] \oplus M[3i-1] \oplus M[3i] \oplus 2^{i-1}L_4$
  $D[i] \leftarrow P_4\big(Y[i]\big)$
end
$S \leftarrow C[1] \oplus \cdots \oplus C[3m] \oplus D[1] \oplus \cdots \oplus D[m]$
$T \leftarrow P_5(S)$
return T

Algorithm 4: The rate-3/4 construction $\mathtt{PMAC3/4}[P_1, P_2, P_3, P_4, P_5]$

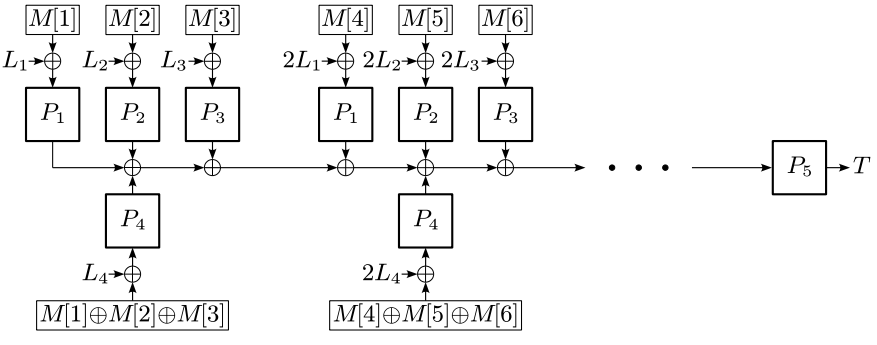


Fig. 3. A pictorial representation of the rate-3/4 construction

5 Smaller-Rate Versions

We generalize our basic version to obtain rate-3/4, 4/5, etc. constructions. To do this, we use the idea from [19]. See Algorithm 4 and Fig. 3 for the definition of rate-3/4 version. The key idea is to make the sum of three blocks, rather than two. Similarly, the sum of four blocks would yield the rate-4/5 construction.

The three-sum version indeed has a smaller rate of 3/4 but at the same time introduces a couple of problems. One is inefficiency in padding; depending on the message length, one might have to pad relatively large number of zeros to make the number of blocks divisible by three. The other is the larger number of keys and larger memory size; note that now we have to store four mask values, L_1, L_2, L_3 and L_4.

However, we still have the same security level as the rate-2/3 construction, as shown in the following theorem. Its proof is similar to the rate-2/3 case and hence omitted.

Theorem 2. *The algorithm* $\mathtt{PMAC3/4}[E_{K_1}, \ldots, E_{K_5}]$ *is a secure PRF on the assumption that the underlying block cipher* E *is a secure PRP. More precisely, we have*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathtt{PMAC3/4}[E_{K_1},\ldots,E_{K_5}]}(t, q, \ell, \sigma) \leq \frac{q^2}{2^n} + \frac{\ell\sigma q}{2^{2n}} + 5\,\mathrm{Adv}^{\mathrm{prp}}_{E}(t', \sigma),$$

where t' *is* t *plus the time complexity to compute the* E *algorithm* σ *times.*

6 Concluding Remarks

We have provided a new PMAC variant whose security bound is of the form $O(q^2/2^n + \ell\sigma q/2^{2n})$. The bound becomes ℓ-free as $O(q^2/2^n)$ under the condition $\ell \leq 2^{n/2}$. Ideally, we would like to obtain $O(q^2/2^n)$ under the condition $\ell \leq 2^n$, but no such constructions (without randomization) seem to be known.

Our construction achieves rate 2/3, 3/4, etc.. Ideally, we would like to have a rate-1 construction having a similar security bound. Again, no such constructions (without randomization) seem to be known.

Finally, we make a remark about the number of keys. Our construction requires at least four independent keys. It would certainly be desirable if one could reduce the number of block-cipher keys. It is of both practical and theoretical interest to develop techniques to reduce the number of keys in this kind of situation.

Acknowledgments. The author would like to thank CT-RSA 2012 PC members and reviewers for accurate and helpful feedback.

References

1. Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 15–28. Springer, Heidelberg (1995)
2. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
3. Bellare, M., Pietrzak, K., Rogaway, P.: Improved Security Analyses for CBC MACs. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 527–545. Springer, Heidelberg (2005)
4. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
5. Bernstein, D.J.: How to stretch random functions: The security of Protected Counter Sums. J. Cryptology 12(3), 185–192 (1999)
6. Black, J.A., Rogaway, P.: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 197–215. Springer, Heidelberg (2000)

7. Black, J.A., Rogaway, P.: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 384–397. Springer, Heidelberg (2002)
8. Dodis, Y., Pietrzak, K.: Improving the Security of MACs via Randomized Message Preprocessing. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 414–433. Springer, Heidelberg (2007)
9. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
10. JTC1. ISO/IEC 9797-1:1999 Information technology—Security techniques—Message Authentication Codes (MACs)—Part 1: Mechanisms using a block cipher (1999)
11. Kurosawa, K., Iwata, T.: TMAC: Two-Key CBC MAC. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 33–49. Springer, Heidelberg (2003)
12. Minematsu, K., Matsushima, T.: New Bounds for PMAC, TMAC, and XCBC. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 434–451. Springer, Heidelberg (2007)
13. Nandi, M.: Fast and Secure CBC-Type MAC Algorithms. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 375–393. Springer, Heidelberg (2009)
14. Nandi, M.: A Unified Method for Improving PRF Bounds for a Class of Blockcipher Based MACs. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 212–229. Springer, Heidelberg (2010)
15. Petrank, E., Rackoff, C.: CBC MAC for real-time data sources. J. Cryptology 13(3), 315–338 (2000)
16. Preneel, B., van Oorschot, P.C.: MDx-MAC and Building Fast MACs from Hash Functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
17. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
18. Sarkar, P.: Pseudo-random functions and parallelizable modes of operations of a block cipher. IEEE Transactions on Information Theory 56(8), 4025–4037 (2010)
19. Yasuda, K.: Multilane HMAC—Security beyond the Birthday Limit. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 18–32. Springer, Heidelberg (2007)
20. Yasuda, K.: The Sum of CBC MACs Is a Secure PRF. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 366–381. Springer, Heidelberg (2010)
21. Yasuda, K.: A New Variant of PMAC: Beyond the Birthday Bound. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 596–609. Springer, Heidelberg (2011)

Boomerang Attacks on Hash Function Using Auxiliary Differentials

Gaëtan Leurent and Arnab Roy

Université du Luxembourg and SnT

Abstract. In this paper we study boomerang attacks in the chosen-key setting. This is particularly relevant to hash function analysis, since many boomerang attacks have been described against ARX-based designs.

We present a new way to combine message modifications, or auxiliary differentials, with the boomerang attack. We show that under some conditions, we can combine three independent paths instead of two for the classical boomerang attack. Our main result is obtained by applying this technique to round-reduced Skein-256, for which we show a distinguisher on the keyed permutation with complexity only 2^{57}, and a distinguisher on the compression function with complexity 2^{114}. We also discuss application of the technique to Skein-512 and show some problems with the paths used in previous boomerang analysis of Skein-512.

Keywords: Hash function, SHA-3 competition, chosen-key, Skein, Threefish, boomerang attack, higher order differential, zero-sum.

1 Introduction

The boomerang attack was proposed by Wagner in 1999 [16] as a cryptanalysis technique against block ciphers. This clever idea allows to combine short differential paths for the top half and the bottom half of a cipher, instead of using a long differential path for the full cipher.

Recently, this idea has been applied to hash function building blocks, as part of the new hash function results inspired by the SHA-3 competition. In [4] Biryukov *et al.* proposed boomerang distinguishers on compression functions and applied it to round-reduced BLAKE. Mendel and Lamberger [11] also independently proposed a boomerang attack on the compression function of SHA-2. More recently at SAC 2011, Yu Sasaki [14] gave a boomerang distinguisher on the full compression function of HAVAL. Boomerang distinguishers have also been applied to Skein/Threefish [1,5].

Another related work by Joux and Peyrin [7] studies boomerangs in the context of hash functions. However this result does not try to build a boomerang property for a hash function, but only uses *auxiliary differential paths*, which are related to the boomerang idea, as a tool for message modifications.

The SHA-3 competition [13] is now at the final phase with 5 remaining hash function candidates; and Skein is one of them. It is one of the two ARX (Addition-Rotation-Xor) designs amongst those candidates.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 215–230, 2012.

The most successful attack proposed against Skein is the rotational rebound attack [10], by Khovratovich *et al.*, reaching 57 out of 72 rounds. This work improves upon the rotational cryptanalysis technique [9] which reached 42 rounds for Skein-512 and 39 rounds for Skein-256. Those results are based on rotation-invariant constants in the key schedule. However for the final round of the SHA-3 competition, Skein has been tweaked [6] to avoid those rotation invariant constants, and this technique is no longer applicable.

Various other techniques have also been applied to Skein, which do not depend on rotation-invariant constants. Su *et al.* [15] gave near collisons on 24 rounds for a cost of 2^{60} and 2^{230} compression function calls on Skein-256 and Skein-512 respectively. Aumasson *et al.* [1] also used a boomerang attack to launch a key recovery attack on Threefish-512 for 32, 34 and 35 rounds. In [5] Chen and Jia proposed a bommerang distinguisher with complexity 2^{189} on 32-rounds of Threefish-512, and used it to mount a key-recovery attacks on 33 and 34 rounds, with complexity $2^{324.6}$ and $2^{474.4}$, respectively. However, we show in Section 5.2 that the paths used in this attack are in fact incompatible. Another recent result by Yu *et al.* gives semi-free start near collision for up to 32 rounds of Skein-256 [18] with complexity 2^{105}.

Our Contributions. We study boomerang distinguishers on round-reduced Skein-256. The analysis is based on high probability related-key differential trails in Threefish (probability 1, 2^{-6}, and 2^{-39} for 8, 12, and 16 rounds respectively), like previous analysis [1,5,15].

Our main contribution is a technique using auxiliary differentials, which allows to skip some rounds in the middle of the boomerang in a chosen-key setting. This is similar to previous works using message modifications (*e.g.* on Skein-512 [18,10]) but we use it in a boomerang setting. When applied to the 32-round attack on Skein, we can avoid 8 rounds in the middle, and this results in a significant complexity improvement. Moreover, since the complexity is relatively low, we can experimentally measure the amplification effect, by implementing the attack on 28 rounds. This results in a boomerang distinguisher with complexity 2^{57} for the keyed-permutation (*i.e.* for Threefish-256). When applied to the compression function with the feed-forward, we show that our attack has complexity at most 2^{114}, but we cannot measure experimentally the full effect of the amplification, so we expect the actual complexity to be significantly lower. A summary of our results is given in Table 1.

Additionally, we also discuss the hypothesis of independence between the boomerang paths for ARX primitives, and give an example of previous work where the hypothesis is not valid, and the paths are in fact incompatible.

2 The Boomerang Attack

The boomerang attack was introduced by David Wagner in 1999 [16] against block ciphers, and the initial idea has been developed through many later results,

Table 1. Summary of the attacks on the compression function (CF) and the keyed permutation (KP) of Skein. We only mention results which are independent of the constants used Skein (*i.e.* which apply to the round-3 version).

Attack	CF/KP	Rounds	CF/KP calls	Reference
Near collisions (Skein-256)	CF	24	2^{60}	[15]
Boomerang dist. (Threefish-512)	KP	32	2^{189}	[5]
Key Recovery (Threefish-512)	KP	33	$2^{324.6}$	[5]
Key Recovery (Threefish-512)	KP	34	$2^{474.4}$	[5]
Near collisions (Skein-256)	CF	32	2^{105}	[18]
Key Recovery (Threefish-512)	KP	32	2^{312}	[1]
Key Recovery (Threefish-512)	KP	35	2^{478}	[1]
Boomerang dist. (Skein-256)	CF and KP	24	2^{18}	Sec. 4.2
Boomerang dist. (Threefish-256)	KP	28	2^{21}	Sec. 4.2
Boomerang dist. (Skein-256)	CF	28	2^{24}	Sec. 4.2
Boomerang dist. (Threefish-256)	KP	32	2^{57}	Sec. 4.2
Boomerang dist. (Skein-256)	CF	32	2^{114}	Sec. 4.2

including [16,8,2,3,4,11]. In this section, we go through the main results of those papers, in order to explain the techniques needed used in our attack on Skein.

The main idea of the boomerang attack is to consider a block cipher as a composition of two sub-ciphers, and to use an encryption oracle as well as a decryption oracle in order to build differential pair for each sub-cipher independently. Given a permutation f that can be decomposed into two sub-permutations f_a and f_b with $f = f_b \circ f_a$ (e.g. a block cipher), one first identifies some high probability differentials[1] $\alpha \to \alpha'$ with probability p_a for f_a and $\gamma \to \gamma'$ with probability p_b for f_b, relative to a group operation $+$ (in practice, the group operation is either the exclusive or $\oplus$, or the modular addition $\boxplus$).

The attacker selects two plain-texts $P^{(0)}$ and $P^{(1)}$ with $P^{(1)} = P^{(0)} + \alpha$, and requests the corresponding cipher-texts $C^{(0)}$ and $C^{(1)}$. Then he builds the cipher-texts $C^{(2)} = C^{(0)} + \gamma'$ and $C^{(3)} = C^{(1)} + \gamma'$, and requests the corresponding plain-texts $P^{(2)}$ and $P^{(3)}$. This is illustrated by Figure 1. With this construction, we expect that $P^{(3)} = P^{(2)} + \alpha$ with probability $p_a^2 p_b^2$. This comes from the following observations:

(i) With probability p_a, $(P^{(0)}, P^{(1)})$ is a good pair for the differential $\alpha \to \alpha'$ in f_a, and we have $X^{(1)} = X^{(0)} + \alpha'$, where $X^{(i)} = f_a(P^{(i)})$.

(ii) With probability p_b^2, $(C^{(0)}, C^{(2)})$ and $(C^{(1)}, C^{(3)})$ are good pairs for the differential $\gamma' \to \gamma'$ in f_b^{-1}, and we have $X^{(2)} - X^{(0)} = X^{(3)} - X^{(1)} = \gamma$, where $X^{(i)} = f_b^{-1}(C^{(i)})$.

(iii) If *(i)* and *(ii)* are satisfied, then we have

$$X^{(3)} - X^{(2)} = (X^{(3)} - X^{(1)}) - (X^{(2)} - X^{(0)}) + (X^{(1)} - X^{(0)}) = \alpha'.$$

[1] We use this to denote $\Pr_{x,k}\left[f_a(x+\alpha) = f_a(x) + \alpha'\right] = p_a$.

With probability p_a, $(X^{(2)}, X^{(3)})$ is a good pair for the differential $\alpha' \to \alpha$ in f_a^{-1}, and we have $P^{(3)} = P^{(2)} + \alpha$.

This basic attack gives a distinguisher for f, and it can be extended to a key-recovery attack using partial encryption/decryption.

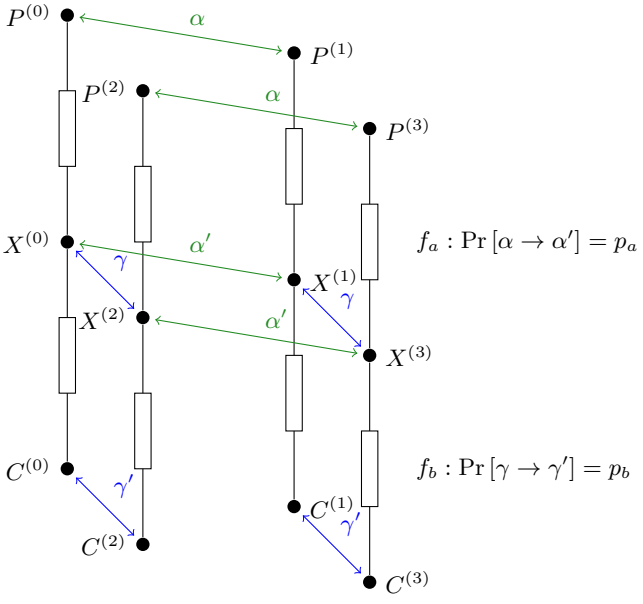


Fig. 1. The boomerang attack

Boomerang attack are particularly efficient on ARX-like designs because the probability of differential paths drops quickly when the number of rounds grows. It is usually possible to find good differential for a few rounds, but extending them leads to more diffusion and very bad probabilities. More generally, if we denote the probability of the best differential for function f by $\mathrm{bp}(f)$, then the boomerang attack is better than classical differential attack if:

$$\mathrm{bp}(f_a)^2 \,\mathrm{bp}(f_b)^2 > \mathrm{bp}(f_b \circ f_a)$$

2.1 Amplified Probabilities

We can compute a better estimate of the complexity of a boomerang attack if we remark that we don't actually need to specify the differences α' and γ. As long as the two pairs $(C^{(0)}, C^{(2)})$ and $(C^{(1)}, C^{(3)})$ reach the same difference $X^{(2)} - X^{(0)} = X^{(3)} - X^{(1)}$, the boomerang attack will work. We can compute

the complexity by summing over all possible α' and γ, which is equivalent to replacing the probabilities p_a and p_b by the following values:

$$\hat{p}_a = \sqrt{\sum_{\alpha'} \Pr[\alpha \to \alpha']} \qquad \hat{p}_b = \sqrt{\sum_{\gamma} \Pr[\gamma \to \gamma']}$$

These are sometimes called *amplified* probabilities, but this is unrelated to the amplified bommerang attack of [8].

We can further improve the complexity by considering two independent differentials $\alpha_0 \to \alpha'_0$ and $\alpha_1 \to \alpha'_1$ in f_a, and $\gamma_0 \to \gamma'_0$ and $\gamma_1 \to \gamma'_1$ in f_b. The paths can be used for a boomerang attack as long as $\alpha'_1 - \alpha'_0 = \gamma_1 - \gamma_0$, as shown in [16].

In practice, the amplified probabilities are often estimated experimentally with random values.

2.2 Related-Key Boomerang

The boomerang attack can also be used with related-key differentials, instead of fixed-key differentials, as shown in [3]. In this case, we use a differential[2] $\alpha, \alpha_k \to \alpha'$ with probability p_a for f_a, and $\beta, \beta_k \to \beta'$ with probability p_b for f_b.

Starting from a random plain-text $P^{(0)}$, we compute

$$\begin{aligned} P^{(1)} &= P^{(0)} + \alpha \\ C^{(0)} &= f(P^{(0)}, k) & C^{(1)} &= f(P^{(1)}, k + \alpha_k) \\ C^{(2)} &= C^{(0)} + \beta' & C^{(3)} &= C^{(1)} + \beta' \\ P^{(2)} &= f^{-1}(C^{(2)}, k + \beta_k) & P^{(3)} &= f^{-1}(C^{(3)}, k + \alpha_k + \beta_k) \end{aligned}$$

and we obtain $P^{(3)} = P^{(2)} + \alpha$ with probability $p_a^2 p_b^2$.

2.3 Application to the Known-Key Setting

In a known-key or chosen-key setting, a boomerang property can be used to distinguish a given permutation from a random one, as first used in [4] and [11]. The boomerang attack can generate quartets $(C^{(i)}, P^{(i)})_{i=0}^{3}$ with:

$$\begin{aligned} C^{(i)} &= f(P^{(i)}) \\ P^{(1)} - P^{(0)} = P^{(3)} - P^{(2)} &= \alpha \qquad C^{(2)} - C^{(0)} = C^{(3)} - C^{(1)} = \gamma' \end{aligned} \tag{1}$$

Alternatively, we can consider the boomerang as a zero-sum property [4], or higher-order differential collision [11] since a quartet satisfies:

$$\begin{aligned} \Delta P^{(i)} &= (P^{(3)} - P^{(2)}) - (P^{(1)} - P^{(0)}) = 0 \\ \Delta C^{(i)} &= (C^{(3)} - C^{(2)}) - (C^{(1)} - C^{(0)}) = (C^{(3)} - C^{(1)}) - (C^{(2)} - C^{(0)}) = 0 \end{aligned}$$

[2] We use this to denote $\Pr_{x,k}[f_a(x + \alpha, k + \alpha_k) - f_a(x, k) = \alpha'] = p_a$.

Moreover, in a known-key or chosen-key setting, it is possible to start from the middle. First one selects some values $X^{(0)}, X^{(1)}, X^{(2)}, X^{(3)}$ with $X^{(2)} - X^{(0)} = X^{(3)} - X^{(1)} = \gamma$ and $X^{(1)} - X^{(0)} = X^{(3)} - X^{(2)} = \alpha'$; then he compute $P^{(i)} = f_a^{-1}(X^{(i)})$ and $C^{(i)} = f_b(X^{(i)})$. This allows to select specific $X^{(i)}$'s that satisfy the paths with better probability that a random quartet.

For an n-bit random permutation, the generic complexity for obtaining a quartet satisfying (1) with fixed α, γ' is 2^n. However, if only the difference $(P^{(3)} - P^{(2)}) = (P^{(1)} - P^{(0)}) = \alpha$ is fixed, the complexity is only $2^{n/2}$. If we only want $\Delta P^{(i)} = 0$ and $\Delta C^{(i)} = 0$, the complexity is lower bounded by $2^{n/3}$, but the best known attack still takes time $2^{n/2}$.

2.4 Application to Hash Function

Boomerang attacks have been applied to hash function, in order to attack some of the components of the design. A bommerang attack can readily be applied to the block-cipher or permutation inside most of the designs. It can also be extended to a distinguisher against the compression function of most block-cipher based designs, as shown in [11] and [4]. For instance let us consider a compression function following the MMO construction: $\mathrm{CF}(h, m) = E_h(m) + m$, and a quartet $P^{(i)}, C^{(i)}$ for the block cipher E under the related keys $K^{(i)}$. The quartet satisfies:

$$\begin{aligned} C^{(i)} &= E_{K^{(i)}}(P^{(i)}) \\ \Delta K^{(i)} &= (K^{(3)} - K^{(2)}) - (K^{(1)} - K^{(0)}) = 0 \\ \Delta P^{(i)} &= (P^{(3)} - P^{(2)}) - (P^{(1)} - P^{(0)}) = 0 \\ \Delta C^{(i)} &= (C^{(3)} - C^{(1)}) - (C^{(2)} - C^{(0)}) = 0. \end{aligned}$$

Moreover, we have

$$\begin{aligned} &\Delta\,\mathrm{CF}(K^{(i)}, P^{(i)}) \\ &\quad = \left[\mathrm{CF}(K^{(3)}, P^{(3)}) - \mathrm{CF}(K^{(2)}, P^{(2)})\right] - \left[\mathrm{CF}(K^{(1)}, P^{(1)}) - \mathrm{CF}(K^{(0)}, P^{(0)})\right] \\ &\quad = \left[(C^{(3)} + P^{(3)}) - (C^{(2)} + P^{(2)})\right] - \left[(C^{(1)} + P^{(1)}) - (C^{(0)} + P^{(0)})\right] \\ &\quad = (C^{(3)} - C^{(1)}) - (C^{(2)} - C^{(0)}) + (P^{(3)} - P^{(2)}) - (P^{(1)} - P^{(0)}) = 0 \end{aligned}$$

This is a zero-sum property for the compression function. For an n-bit random compression function the best known attack to build a quartet with a zero-sum output takes time $2^{n/3}$ using Wagner's generalized birthday attack [17]. If we also want the inputs to be a zero-sum, the best known attack takes time $2^{n/2}$.

3 Boomerang Attack Using Auxiliary Differentials

The main idea of our attack is to use auxiliary paths as message modifications. This idea has already be applied to hash function cryptanalysis by Joux and

Peyrin in [7], in the context of a classical differential attack. Here, we apply it to the boomerang setting, with related-key paths. The main idea is to build boomerang quartet in an inside-out fashion, and to use auxiliary paths to efficiently generate values in the middle, so that they conform to several rounds.

We consider a function f that can be decomposed into three sub-functions $f = f_c \circ f_b \circ f_a$, and we consider a differential in each of those sub-functions:

- for f_a, we use a differential $\alpha \to \alpha'$ with probability p_a
- for f_b, we use a set $\mathcal{B}$ of b differentials $\beta_j \to \beta'_j$ with probability p_b
- for f_c, we use a differential $\gamma \to \gamma'$ with probability p_c

We describe the idea with fixed-key differential for simplicity, but it works in the same way with related key differentials. We start with a boomerang quartet $(U^{(0)}, U^{(1)}, U^{(2)}, U^{(3)}) \to (V^{(0)}, V^{(1)}, V^{(2)}, V^{(3)})$ for f_b, with

$$U^{(1)} = U^{(0)} + \alpha' \quad U^{(3)} = U^{(2)} + \alpha' \quad V^{(2)} = V^{(0)} + \gamma \quad V^{(2)} = V^{(1)} + \gamma$$

Using an auxiliary path $\beta_j \to \beta'_j$, we construct $U_*^{(i)} = U^{(i)} + \beta_j$. With probability p_b^4, we obtain a new quartet $(U_*^{(0)}, U_*^{(1)}, U_*^{(2)}, U_*^{(3)}) \to (V_*^{(0)}, V_*^{(1)}, V_*^{(2)}, V_*^{(3)})$, where $V_*^{(i)} = V^{(i)} + \beta'_j$. Then, we have

$$U_*^{(1)} = U_*^{(0)} + \alpha' \quad U_*^{(3)} = U_*^{(2)} + \alpha' \quad V_*^{(2)} = V_*^{(0)} + \gamma \quad V_*^{(2)} = V_*^{(1)} + \gamma$$

We compute the plain-texts and cipher-texts corresponding to these values, and with probability $p_a^2 \cdot p_c^2$, this will result in a boomerang quartet for f, as shown in Figure 2.

If the initial cost to build a quartet for f_b is C, then we can build a quartet for the full f with complexity:

$$\frac{1}{p_a^2 p_c^2}\left(\frac{C}{b \cdot p_b^4} + 1\right)$$

For the application to Skein, we use properties of the key-schedule to build a set of related-key differentials β_k with probability 1. This results in $C \ll b \cdot p_c^4$, and we essentially skip rounds in the middle of the permutation for free.

4 Application to Skein

Brief Description of Skein. The compression function of Skein is based on the block cipher Threefish. Let $U_{r,i}$ be the ith word of the encrypted state after r rounds and n_w be the number of words in a state. Then for each round we have

$$V_{r,i} = \begin{cases} U_{r,i} + K_{r/4,i} & \text{if } r \bmod 4 = 0 \\ U_{r,i} & \text{otherwise} \end{cases}$$

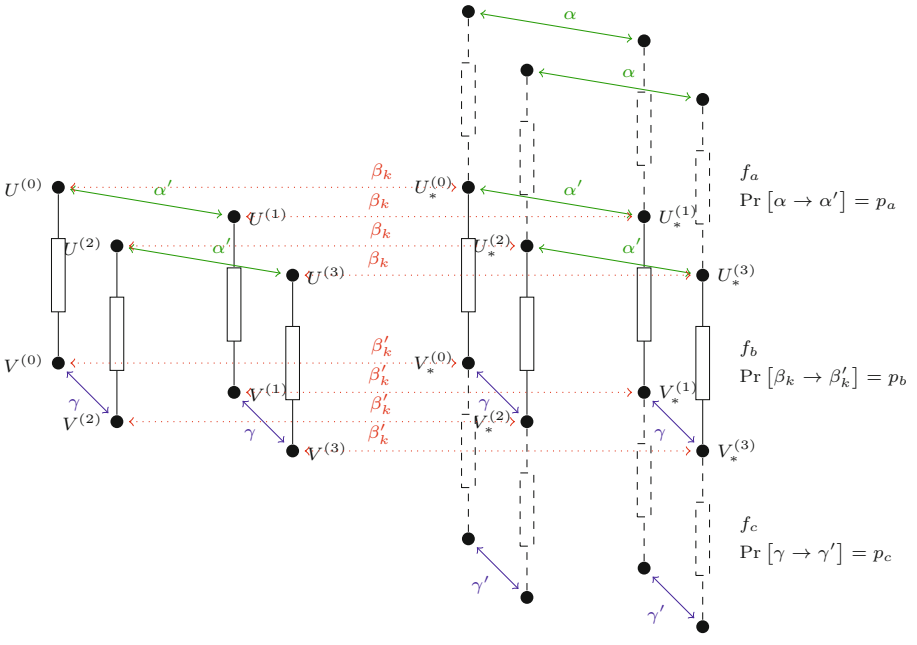


Fig. 2. Using auxiliary paths in a boomerang distinguisher

where $K_{r/4,i}$ is the ith word of the round key at round $r/4$. The state $U_{r+1,i}$ (for $i = 0, 1, .., n_w$) after round $r + 1$ is obtained from $V_{r,i}$ by applying a MIX transformation and a permutaion of n_w words as following:

$$\begin{array}{lll} (X_{r,2k}, X_{r,2k+1}) & := \texttt{MIX}_{r,k}(V_{r,2k}, V_{r,2k+1}) & \text{for } k = 0, 1, .., n_w/2 \\ U_{r+1,i} & := X_{r,\sigma(i)} & \text{for } i = 0, 1, .., n_w \end{array}$$

where σ is a permutaion specified in [6] and $(c, d) = \texttt{MIX}_{r,k}(a, b)$ is described as

$$\begin{aligned} c &= (a + b) \bmod 2^{64} \\ d &= (b \lll R_{r \bmod 8,k}) \oplus c \end{aligned}$$

The rotations $R_{r \bmod 8,k}$ are speicifed in [6]. The key scheduling algorithm of Threefish produces 18 round keys from a tweak (T_0, T_1) and a key as following

$$\begin{array}{ll} K_{l,i} = K_{(l+i) \bmod (n_w+1)} & \text{for } i = 0, 1, .., n_w - 4 \\ K_{l,i} = K_{(l+i) \bmod (n_w+1)} + T_{l \bmod 3} & \text{for } i = n_w - 3 \\ K_{l,i} = K_{(l+i) \bmod (n_w+1)} + T_{l \bmod 3} & \text{for } i = n_w - 2 \\ K_{l,i} = K_{(l+i) \bmod (n_w+1)} + l & \text{for } i = n_w - 1 \end{array} .$$

where $K_{n_w} = C_{240} \oplus \bigoplus_{i=0}^{n_w-1} K_i$ with C_{240} a constant specified in [6], and $T_2 = T_0 \oplus T_1$. The compression function F for Skein is given as $F = E_{CV,T}(M) \oplus M$. For Skein-256 $n_w = 4$ and word size is 8 byte. We use the notation $k^{(l)}$ to denote the expanded key used at round $4l$, *i.e.* $k^{(l)} = K_{l,0}, K_{l,1}, \ldots K_{l,n_w-1}$.

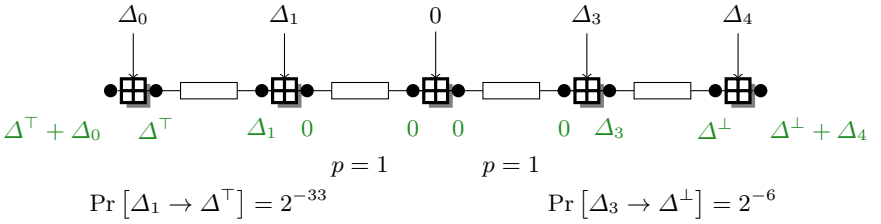


Fig. 3. Linearized differential path for Skein

Table 2. Subkey differential trails

Round	Subkey Trail$_1$: K_3, K_4, T_0, T_2				Round	Subkey Trail$_2$: K_2, K_3, T_0, T_1			
0	K_0	$K_1 + \underline{T_0}$	$K_2 + T_1$	$\underline{K_3} + 0$	16	K_4	$K_0 + \underline{T_1}$	$K_1 + T_2$	$\underline{K_2} + 4$
4	K_1	$K_2 + T_1$	$\underline{K_3} + \underline{T_2}$	$\underline{K_4} + 1$	20	K_0	$K_1 + T_2$	$\underline{K_2} + \underline{T_0}$	$\underline{K_3} + 5$
8	K_2	$\underline{K_3} + \underline{T_2}$	$\underline{K_4} + \underline{T_0}$	$K_0 + 2$	24	K_1	$\underline{K_2} + \underline{T_0}$	$\underline{K_3} + \underline{T_1}$	$K_4 + 6$
12	$\underline{K_3}$	$\underline{K_4} + \underline{T_0}$	$K_0 + T_1$	$K_1 + 3$	28	$\underline{K_2}$	$\underline{K_3} + \underline{T_1}$	$K_4 + T_2$	$K_0 + 7$
16	$\underline{K_4}$	$K_0 + T_1$	$K_1 + \underline{T_2}$	$K_2 + 4$	32	$\underline{K_3}$	$K_4 + T_2$	$K_0 + \underline{T_0}$	$K_1 + 8$

4.1 Round-Reduced Differential Trails in Skein-256

Due to the key schedule of Skein, it is possible to build differential trails over 8 rounds with probability one, using a difference in the tweak T. To do this, we just use a key difference and tweak difference that cancel each other at a given round r, and we compute the corresponding key differences for rounds $r+4$ and $r-4$. By injecting this difference in the state, we obtain an 8-round path.

In order to achieve the best probability, we use a difference Δ_{msb} on the most significant bit of both tweaks used at round r, and on the corresponding keys. This results in a 12-round path with probability 2^{-6} and a 16-round path with probability 2^{-43} (we don't consider the key addition for those probabilities). The path is shown in Figure 3. For a boomerang attack on 32-round Skein, we use this with $r = 8$ and $r = 24$, and the corresponding key-differential as shown in Table 2. Previous analysis of Skein [1,5,15,18] are based on the same trails.

For our attack, we also need a set of auxiliary paths. We build this set using the same 8-round paths with probability one, but we do not restrict ourselves to a difference on the most significant bit. We can set the tweak T to an arbitrary value, and recompute the key in order the have the same expanded key $k^{(4)}$ at rounds 16. This gives a set of 2^{128} paths with probability one.

4.2 Description of the Attack on Skein-256

Our attack on skein is similar to a boomerang attack on 32 rounds using two 16-round trails, but we build a valid quartet starting from the middle, and we use auxiliary trails to avoid paying the probabilities of 8 rounds in the middle. We proceed with three consecutive steps, as shown by Figure 5, page 230.

First Step. The first part of the attack considers rounds 16 to 20. We try to build a quartet $u^{(i)} = R(t^{(i)})$ with

$$t^{(0)} \oplus t^{(1)} = t^{(2)} \oplus t^{(3)} = \Delta^{\perp} \oplus \Delta_4 \tag{2}$$

$$u^{(0)} \oplus u^{(2)} = u^{(1)} \oplus u^{(3)} = \Delta_1 \tag{3}$$

One possible way to build such a quartet is to start with a set of $t^{(i)}$ that satisfies (2) and $t^{(0)} \oplus t^{(2)} = t^{(1)} \oplus t^{(3)} = \Delta^{\perp} \oplus \Delta_4$, and to compute the corresponding $u^{(i)} = R(t^{(i)})$. The quartet will satisfy (3) with probability 2^{-66}, but we can fix some bits of the state in order to improve this complexity.

Actually, it is more efficient to follow the steps of a boomerang attack:

- start from a pair $t^{(0)}, t^{(1)}$ with $t^{(0)} \oplus t^{(1)} = \Delta^{\perp} \oplus \Delta_4$;
- compute $u^{(0)} = R(t^{(0)})$, $u^{(1)} = R(t^{(1)})$, $u^{(2)} = u^{(0)} \oplus \Delta_1$, $u^{(3)} = u^{(1)} \oplus \Delta_1$;
- compute $t^{(2)} = R^{-1}(u^{(2)})$ and $t^{(3)} = R^{-1}(u^{(3)})$; check whether (2) holds.

Using this procedure allows us to benefit from amplified probabilities, since we do not specify the path from $(u^{(0)}, u^{(2)})$ to $(t^{(0)}, t^{(2)})$ and from $(u^{(1)}, u^{(3)})$ to $(t^{(1)}, t^{(3)})$, respectively, we only check that the differences are the same. Experimentations show that this step costs around 2^{18}.

Second Step. The second part of the attack concerns rounds 12 to 16. We start with a quartet $u^{(i)} = R(t^{(i)})$ satisfying (2) and (3), and we want to extend it with $s^{(i)} = R^{-1}(t^{(i)} - k_4^{(i)})$ so that

$$s^{(0)} \oplus s^{(1)} = s^{(2)} \oplus s^{(3)} = \Delta_3. \tag{4}$$

The main idea is to use the key injection at round 16 in order to randomize the state, until we find pairs that follows the differential $\Delta^{\perp} \to \Delta_3$. First we select the keys that result in:

$$(t^{(0)} - k_4^{(0)}) \oplus (t^{(1)} - k_4^{(1)}) = (t^{(2)} - k_4^{(2)}) \oplus (t^{(3)} - k_4^{(3)}) = \Delta^{\perp} \qquad \text{with}$$
$$k_4^{(0)} \oplus k_4^{(1)} = k_4^{(2)} \oplus k_4^{(3)} = \Delta_4 \qquad \text{and} \qquad k_4^{(0)} \oplus k_4^{(2)} = k_4^{(1)} \oplus k_4^{(3)} = \Delta_0.$$

We can find the suitable solution by solving a simple system of additions and xors. Then, we compute the corresponding $s^{(i)}$, and we check whether (4) is satisfied. On average, we expect this step to cost 2^{12}. According to our experimentations, however, there seem to be some dependency between the paths, and the average cost is about 2^{18}.

This step can be seen as an application of the technique of Section 3. We use a trivial related-key differential where the key difference just cancels the state difference in order to extend a 4-round quartet into 8-round quartets.

Third Step. This is the main step of the attack, following the ideas of Section 3. We start with a quartet $u^{(i)} = R(R(s^{(i)}) + k_4^{(i)})$, and we use probability-1 differentials to build many more quartets, until the top and bottom paths are satisfied.

The best result are achieved using the modular difference, because the key-additions are modular additions. Note that we include the initial and final key-addition in our 32-round reduced Threefish/Skein. More precisely, for each quartet generated for rounds 12–16 verifying (3) and (4), we compute the corresponding plain-text and cipher-text and we check whether

$$\Delta^{\boxplus} P^{(i)} = (P^{(3)} \boxminus P^{(2)}) \boxminus (P^{(1)} \boxminus P^{(0)}) = 0 \tag{5}$$

$$\Delta^{\boxplus} C^{(i)} = (C^{(3)} \boxminus C^{(2)}) \boxminus (C^{(1)} \boxminus C^{(0)}) = 0 \tag{6}$$

Experimentally, a quartet satisfies (5) with probability 2^{-36} and (6) with probability 2^{-21} (see Appendix A). This gives a distinguisher for the keyed permutation with complexity around 2^{57}. Note that if we do an analysis similar to the one in [5], we would expect this attack to have a complexity of around 2^{95}; the amplification effect detected in practice is much stronger than predicted by [5].

If we want to build a distinguisher for the compression function, we will instead use the xor-difference, because the feed-forward is an xor operation. Therefore, we will check whether:

$$\Delta^{\oplus} P^{(i)} = P^{(0)} \oplus P^{(1)} \oplus P^{(2)} \oplus P^{(3)} = 0 \tag{7}$$

$$\Delta^{\oplus} C^{(i)} = C^{(0)} \oplus C^{(1)} \oplus C^{(2)} \oplus C^{(3)} = 0 \tag{8}$$

Experimentally, we find that (8) is verified with probability 2^{-24}. The probability for (7) is too low to check experimentally, but we can estimate it from the probability of (5): a quartet satisfying (5) is composed of two pairs of plain-text with $(P^{(1)} \boxminus P^{(0)}) = (P^{(3)} \boxminus P^{(2)}) = \Delta$, where Δ has weight around 34. For each active position, there is a probability $1/3$ that the carry extension in $(P^{(0)}, P^{(1)})$ is the same as in $(P^{(2)}, P^{(3)})$, which leads to:

$$\Pr\left[\Delta^{\oplus} P^{(i)} = 0\right] \geq \Pr\left[\Delta^{\boxplus} P^{(i)} = 0\right] \times (1/3)^{34} \geq 2^{-90}.$$

This gives a distinguisher on the compression function with complexity 2^{114}. In practice we expect the complexity to be significantly lower, because a quartet satisfying (7) does not necessarily satisfy (5).

Attack on 24 and 28 rounds can be build with the same approach.

5 Extensions and Limitations

The technique described in the previous sections can be applied to improve almost any chosen-key boomerang distinguisher. The main limitation is that we need to be able to generate an initial quartet for the middle rounds, similarly to what we do in steps one and two of the attack on Skein-256. We note that any successful boomerang attack does provide such quartets; therefore, as long as a standard boomerang attack works, our improved attack with auxiliary differentials will also work.

However, an often overlooked problem of boomerang attacks is that we need the top and bottom paths to be somehow independent. More precisely, in a standard boomerang attack as depicted by Figure 1, we expect that if a pair $(P_0, P_1) \to (X_0, X_1)$ with is a good pair for f_a (*i.e.* $P_1 = P_0 + \alpha$ and $X_1 = X_0 + \alpha'$), then the pair $(X_0 + \gamma, X_1 + \gamma)$ behaves like a random pair regarding f_a, and will satisfy the differential with probability p_a. However, in practice this may not be the case.

In the following section, we discuss cases where boomerang attacks on ARX design can fail because of this property. This problem was already discussed by Murphy in [12], where he gives examples of non-compatible paths for the DES and the AES. It has also been discussed by Sasaki in [14] for boomerang attacks on Haval.

5.1 Extension to More Rounds

We tried to extend the attack by adding middle rounds, at the bottom of the top path, or at the top of the bottom path. For instance, using a 16-round path for the bottom part should only increase the complexity by a factor of roughly 2^{12}. However, this usually results in incompatible paths, for which no valid quartet exist. In particular linearized paths are incompatible, and we have not been able to build compatible paths.

By studying those incompatible paths, we found that very simple patterns can lead to incompatibilities. Figure 4 gives an example of a pattern that results in incompatible paths. A quartet following those paths would have to satisfy:

$$x^{(0)} \oplus x^{(2)} = x^{(1)} \oplus x^{(3)} = \mathtt{01} \quad y^{(0)} \oplus y^{(2)} = y^{(1)} \oplus y^{(3)} = \mathtt{00} \quad \textit{(Top)} \tag{9}$$

$$x^{(0)} \oplus x^{(1)} = x^{(2)} \oplus x^{(3)} = \mathtt{01} \quad y^{(0)} \oplus y^{(1)} = y^{(2)} \oplus y^{(3)} = \mathtt{01} \quad \textit{(Bottom)} \tag{10}$$

$$x^{(0)} \boxplus y^{(0)} = x^{(1)} \boxplus y^{(1)} \qquad x^{(2)} \boxplus y^{(2)} = x^{(3)} \boxplus y^{(3)} \tag{11}$$

Without loss of generality, we can assume that $\mathrm{lsb}(x^{(0)}) = 0$. This implies $\mathrm{lsb}(x^{(1)}) = 1$ from (10), and $\mathrm{lsb}(x^{(2)}) = 1$ and $\mathrm{lsb}(x^{(3)}) = 0$ from (9). We can deduce $y^{(0)} = y^{(1)} \boxplus 1$ and $y^{(3)} = y^{(2)} \boxplus 1$ from (11). Combined with (10) this yields $\mathrm{lsb}(y^{(0)}) = 1$, $\mathrm{lsb}(y^{(1)}) = 0$, $\mathrm{lsb}(y^{(2)}) = 0$, and $\mathrm{lsb}(y^{(3)}) = 1$. This is incompatible with (9).

This pattern seems to appear very frequently when using linearized paths in ARX designs, and shows that some very natural paths cannot be combined in a boomerang attack.

5.2 Application to Skein-512

By applying our technique to Skein-512, we would expect distinguishers with a similar complexity for the same number of rounds. However, in order to apply the technique, we need to be able to generate quartets for the middle rounds, and we failed to build any such quartet for a 32-round attack. Since a boomerang attack on Skein-512 was presented in [5], we studied the paths used in this attacks, and we found that they are in fact not compatible.

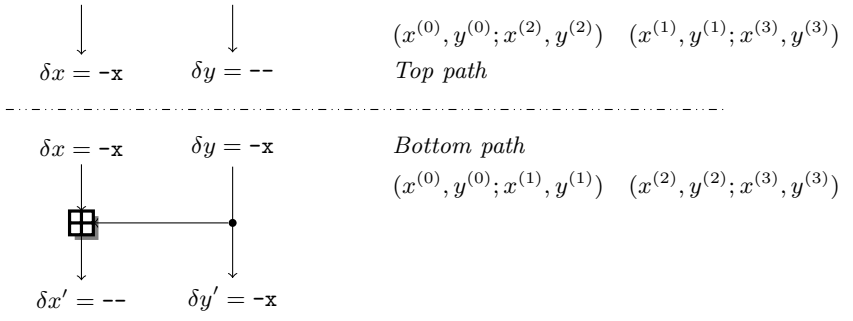


Fig. 4. Example of incompatible paths

Following the notations of [5], the path for rounds 0–16 has a difference $e_{15,5}[10, 39, 49, *64]$. If this path is applied to states $(e^{(0)}, e^{(1)})$ and $(e^{(2)}, e^{(3)})$, this implies that on bit 49, we have:

$$v_{15,5}^{(0)} = 0 \qquad v_{15,5}^{(1)} = 1 \qquad v_{15,5}^{(2)} = 0 \qquad v_{15,5}^{(3)} = 1 \tag{12}$$

Assuming there are no carries, the path for round 16–32 has $e_{16,5}[-34, -50]$ and $e_{16,2}[5, 11, 16, 41, 44, 47]$. Since $e_{16,2}, e_{16,5} = \texttt{MIX}(e_{15,4}, e_{15,5})$, and the rotation used at that step is 56, we have $e_{15,5} = (e_{16,2} \oplus e_{16,5})^{\ggg 56}$. This results in $e_{15,5}$ being active on bits 13, 19, 24, 42, 49, 52, 55, and 58. If this path is applied to states $(e^{(0)}, e^{(2)})$ and $(e^{(1)}, e^{(3)})$, this implies that on bit 49, we have $v_{15,5}^{(0)} \neq v_{15,5}^{(2)}$ and $v_{15,5}^{(0)} \neq v_{15,5}^{(2)}$. This is contradictory with (12).

We tried to fix the paths using a carry extension from bit 34 to 41 in $e_{16,5}$, by using different signs in the paths $(e^{(0)}, e^{(1)})$ and $(e^{(2)}, e^{(3)})$, or by using a carry extension in $e_{15,5}[49]$, but this always ends up with a similar contradiction. However, we note that if the attack of [5] can be fixed, then our technique is expected to yield a distinguisher on Skein-512 with complexity similar to the distinguisher on Skein-256.

6 Conclusions

In this paper we have presented a technique to improve the boomerang attack in the chosen-key setting and applied it to obtain an efficient distinguisher for 32 rounds of the compression function of Skein-256. We also discuss extension of the attack, and application of the technique to Skein-512. This technique can essentially be used to improve any chosen-key boomerang distinguisher. However, we explain that boomerang attack on ARX-design can fail because of incompatible paths. This is not a limitation of the auxiliary paths, but of the underlying boomerang technique.

Acknowledgment. Gaëtan Leurent is supported by the AFR grant PDR-10-022 of the FNR Luxembourg.

References

1. Aumasson, J.P., Calik, C., Meier, W., Ozen, O., Phan, R.C.W., Varici, K.: Improved Cryptanalysis of Skein. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 542–559. Springer, Heidelberg (2009)
2. Biham, E., Dunkelman, O., Keller, N.: The Rectangle Attack - Rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
3. Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
4. Biryukov, A., Nikolić, I., Roy, A.: Boomerang Attacks on BLAKE-32. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 218–237. Springer, Heidelberg (2011)
5. Chen, J., Jia, K.: Improved Related-Key Boomerang Attacks on Round-Reduced Threefish-512. In: Kwak, J., Deng, R.H., Won, Y., Wang, G. (eds.) ISPEC 2010. LNCS, vol. 6047, pp. 1–18. Springer, Heidelberg (2010)
6. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family. Submission to NIST (2008/2010)
7. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
8. Kelsey, J., Kohno, T., Schneier, B.: Amplified Boomerang Attacks against Reduced-Round MARS and Serpent. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (2001)
9. Khovratovich, D., Nikolić, I.: Rotational Cryptanalysis of ARX. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 333–346. Springer, Heidelberg (2010)
10. Khovratovich, D., Nikolić, I., Rechberger, C.: Rotational Rebound Attacks on Reduced Skein. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 1–19. Springer, Heidelberg (2010)
11. Lamberger, M., Mendel, F.: Higher-order differential attack on reduced SHA-256. Cryptology ePrint Archive, Report 2011/037 (2011), http://eprint.iacr.org/
12. Murphy, S.: The return of the cryptographic boomerang. IEEE Transactions on Information Theory 57(4), 2517–2521 (2011)
13. National Institute of Standards and Technology: Cryptographic hash algorithm competition, http://csrc.nist.gov/groups/ST/hash/sha-3/index.html
14. Sasaki, Y.: Boomerang distinguishers on MD4-based hash functions: First practical results on full 5-pass HAVAL. In: SAC (2011)
15. Su, B., Wu, W., Wu, S., Dong, L.: Near-Collisions on the Reduced-Round Compression Functions of Skein and BLAKE. In: Heng, S.H., Wright, R.N., Goi, B.M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 124–139. Springer, Heidelberg (2010)
16. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
17. Wagner, D.: A Generalized Birthday Problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
18. Yu, H., Chen, J., Ketingjia, Wang, X.: Near-collision attack on the step-reduced compression function of Skein-256. Cryptology ePrint Archive, Report 2011/148 (2011), http://eprint.iacr.org/

A Boomerang Quartets for 28 Round Threefish

In this section, we give examples of quartets, to show that the techniques used for the 32-round attack are valid. Table 3 gives is a zero-sum for rounds 4–32 of Threefish, with $\Delta^{\boxplus} P'^{(i)} = 0$ and $\Delta^{\boxplus} C^{(i)} = 0$. Generating such a quartet costs around 2^{21}. Table 4 gives is a zero-sum for rounds 0–28 of Threefish, with $\Delta^{\boxplus} P^{(i)} = 0$ and $\Delta^{\boxplus} C'^{(i)} = 0$. Generating such a quartet costs around 2^{36}.

Table 3. A quartet that satisfies the paths for rounds 4–32

Plain-text before round 4				
$P'^{(0)}$	fe5ab24b9481e005	dcf5504b75b919e5	076e43e18e3a50ce	d31433344b540c75
$P'^{(1)}$	fe5ab24b9481e005	dcf5504b75b919e5	076e43e18e3a50ce	531433344b540c75
$P'^{(2)}$	64309deb8a55633f	71f578e8ddfd6e89	4e299c34006568b0	95847e8860164845
$P'^{(3)}$	64309deb8a55633f	71f578e8ddfd6e89	4e299c34006568b0	15847e8860164845
Key				
$K^{(0)}$	674dfabf537e5a73	92a94934d0ca3e21	90ce87c17d8540d1	ff65c869e8cdadd4
$K^{(1)}$	674dfabf537e5a73	92a94934d0ca3e21	90ce87c17d8540d1	7f65c869e8cdadd4
$K^{(2)}$	674dfabf537e5a73	92a94934d0ca3e21	10ce87c17d8540d1	7f65c869e8cdadd4
$K^{(3)}$	674dfabf537e5a73	92a94934d0ca3e21	10ce87c17d8540d1	ff65c869e8cdadd4
Tweak				
$T^{(0,1)}$	182d916b255ae5e8	5cba243a3b82278e	982d916b255ae5e8	5cba243a3b82278e
$T^{(2,2)}$	982d916b255ae5e8	dcba243a3b82278e	182d916b255ae5e8	dcba243a3b82278e

Table 4. A quartet that satisfies the paths for rounds 0–28

Plain-text before round 0				
$P^{(0)}$	d9d7934ee20a9c9a	d7c7d25a8f42f324	25ac377afcb411bb	424daed3f2425bc1
$P^{(1)}$	d4d82358b1e9945a	58c7c2587f63fb24	25ec3b7b6eb84fb7	c20daed37e421dc5
$P^{(2)}$	0404cce3c56e92df	1887e00caa229acc	5bdad7995f5f036a	a7b69f1a1274d559
$P^{(3)}$	ff055ced954d8a9f	9987d00a9a43a2cc	5c1adb99d1634166	27769f199e74975d
Key				
$K^{(0)}$	8cb950f444a069e3	48380fb03c6b84c6	2034665dbf7fbfb9	59a45c529130786a
$K^{(1)}$	8cb950f444a069e3	48380fb03c6b84c6	2034665dbf7fbfb9	d9a45c529130786a
$K^{(2)}$	8cb950f444a069e3	48380fb03c6b84c6	a034665dbf7fbfb9	d9a45c529130786a
$K^{(3)}$	8cb950f444a069e3	48380fb03c6b84c6	a034665dbf7fbfb9	59a45c529130786a
Tweak				
$T^{(0,1)}$	684e3541ef841667	b3a8cd11bb94bb5d	e84e3541ef841667	b3a8cd11bb94bb5d
$T^{(2,3)}$	e84e3541ef841667	33a8cd11bb94bb5d	684e3541ef841667	33a8cd11bb94bb5d

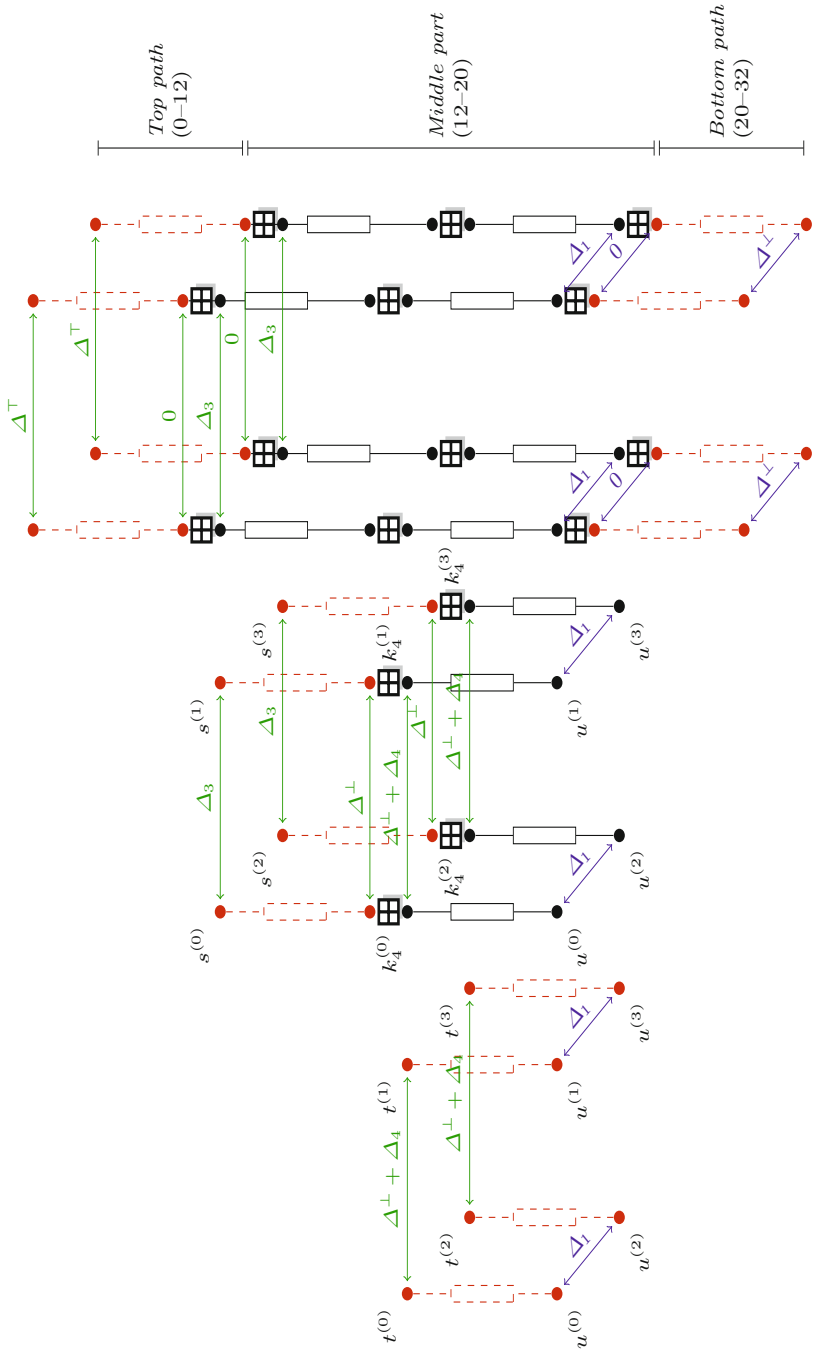


Fig. 5. Overview of the attack, showing the three consecutive steps

Localized Electromagnetic Analysis of Cryptographic Implementations

Johann Heyszl[1], Stefan Mangard[2],
Benedikt Heinz[1], Frederic Stumpf[1], and Georg Sigl[3]

[1] Fraunhofer Research Institution AISEC, Munich, Germany
{johann.heyszl,benedikt.heinz,frederic.stumpf}@aisec.fraunhofer.de
[2] Infineon Technologies AG, Munich, Germany
stefan.mangard@infineon.com
[3] Technische Universität München, EI SEC, Munich, Germany
sigl@tum.de

Abstract. High resolution inductive probes enable precise measurements of the electromagnetic field of small regions on integrated circuits. These precise measurements allow to distinguish the activity of registers on the circuit that are located at different distances to the probe. This location-dependent information can be exploited in side-channel analyses of cryptographic implementations. In particular, cryptographic algorithms where the usage of registers depends on secret information are affected by side-channel attacks using *localized electromagnetic analysis*. Binary exponentiation algorithms which are used in public key cryptography are typical examples for such algorithms. This article introduces the concept of localized electromagnetic analysis in general. Furthermore, we present a case study where we employ a template attack on an FPGA implementation of the elliptic curve scalar multiplication to prove that location-dependent leakage can be successfully exploited. Conventional countermeasures against side-channel attacks are ineffective against location-dependent side-channel leakage. As an effective general countermeasure, we promote that the assignment of registers to physical locations should be repeatedly randomized during execution.

Keywords: Side-channel analysis, electromagnetic, near-field, location-dependent leakage, template attack, FPGA, ECC.

1 Introduction

The physical security of cryptographic implementations is a topic of increasing importance besides the conventional, mathematical security of cryptographic algorithms. Passive side-channel attacks as well as active fault attacks are a major threat. Examples for passive side-channel attacks are timing attacks or simple and differential power attacks [8,9]. Electro-Magnetic (EM) radiation is proportional to the power consumption of devices as well and is a known side-channel since the 1950s [6]. EM side-channels of cryptographic devices and cartography thereof [15] are mostly used to improve conventional side-channel analysis

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 231–244, 2012.

by finding locations where analysis methods lead to better results [18,17,16]. However, it has been demonstrated how the analysis of EM radiation provides advantages compared to the analysis of the power consumption because the analysis can can be limited in terms of location and magnetic field directions [1]. Inductive probes allow high spatial resolutions [6] and can be used to locally restrict measurements of EM radiation [5] if they are placed close to the surface of an integrated circuit. Different distances of hardware registers on an integrated circuit to high-resolution, near-field probes influence EM measurements. This allows to determine locations, where EM radiation is emitted and to coarsely trace data-flows in implementations using local EM measurements [7].

In this article, we present how fine-grained, localized EM measurements can be used to attack cryptographic implementations by exploiting location-dependent side-channel leakage. This is contrary to conventional side-channel attacks which use data-, or operation-dependent leakage. Side-channel analysis methods like collision attacks [20,19] or template attacks [2], which are currently used to exploit data-dependent side-channel leakage, can be applied to exploit location-dependent leakage instead. Potential targets for this localized approach are all cryptographic algorithms where the usage of registers depends on secret information. Binary exponentiation algorithms which are used in modular exponentiations for RSA and in Elliptic Curve Scalar Multiplications (ECSM) are examples of algorithms that are particularly susceptible to location-based side-channel attacks. Hence, we discuss how such algorithms can be attacked in more detail. In a case study, we successfully perform a template attack based on location-dependent EM leakage on an FPGA implementation of the ECSM. Our practical results show how localized electromagnetic analysis leaks sufficient information about the secret to recover it using a single trace. We discuss the insufficiency of previous countermeasures and present a countermeasure which is based on randomizing register locations. As a general countermeasure for affected algorithms we promote that the assignment of certain registers to physical locations should be randomized by swapping their locations at random times.

Our contribution is organized as follows. We explain the principle of localized EM analysis and general attacks based on the localized EM analysis in Sect. 2. In Sect. 3, we apply attacks on binary exponentiation algorithms. Section 4 contains our practical case study. We discuss countermeasures in Sect. 5 and draw conclusions in Sect. 6.

2 Localized EM Analysis

Value changes in CMOS gates lead to dynamic power consumption. The corresponding currents produce concentric magnetic fields around conductors. Variations in the superposed magnetic field of multiple conductors can be measured using inductive sense coils. The field strength is proportional to current changes and decreases with distance to the conductors. In order to measure the magnetic field of small regions of a device, a high spatial resolution, near-field sense coil can be placed close to the surface of an integrated circuit die.

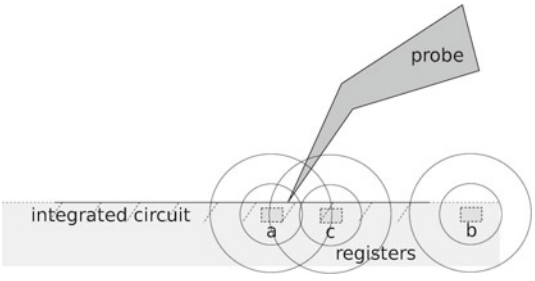


Fig. 1. The distance to the power consuming elements influences the measurement

Cryptographic algorithms use registers to hold data values which are distributed over the integrated circuit. Figure 1 depicts an inductive near-field probe close to the surface of an integrated circuit die with three implemented registers a, b and c in a simplified example. A register is written to by changing control lines and supplying it with a clock signal to update its internal value. All the involved logic gates (e.g., also multiplexers in the datapath) processing the value change consume dynamic power from the supply network which can be measured. A register which is not updated keeps the current value in a feedback mode or is even clock-gated, thus, not consuming dynamic power. The probe in Fig. 1 is closer to register a than to register b. Therefore, activity in register a will lead to greater measured values than activity in register b. However, note that typically, the single bit cells which belong to one multi-bit register will not be located within confined areas. They will be located interspersed. Nevertheless, there are locations on the surface of the die, where the *accumulated distance* of the probe to the power consuming elements of one multi-bit register is shorter than the distance to the elements of another multi-bit register. At those locations it is possible to distinguish, which of the registers has been used based on different signal strengths. Cryptographic algorithms which use their registers differently, depending on the value of the secret, are prone to location-dependent side-channel leakage because the recovery of the fact which registers are used leaks information about the secret.

Location-dependent information leakage can be exploited through attacks based on *localized EM analysis*. Many concepts which exploit data- or operation-dependent side-channel leakage such as SPA attacks in general, side-channel-based collision attacks [20,19], template attacks [2], correlation-based attacks and DPA attacks can be adapted to exploit location-dependent side-channel leakage instead of data-dependent leakage.

In the conventional case of exploiting data-dependent leakage, the unfavorable influence of electronic noise can be reduced through averaging multiple traces with the same processed data. When exploiting location-dependent leakage, the data-dependency of side-channels presents as an unfavorable influence and can be reduced through averaging multiple traces with different processed data.

3 Attacking Binary Exponentiation Algorithms

Algorithm 1. Main loop of an abstract algorithm. Computation sequence and timing are uniform while the register usage depends on secret d.

Input: Secret $d = d_D d_{D-1} ... d_2 d_1$ with $d_i \in \{0, 1\}$

```
1: for i = D downto 1 do
2:    if d_i = 1 then
3:       c ← a
4:       c ← c^2
5:       a ← c
6:    else
7:       c ← b
8:       c ← c^2
9:       b ← c
10:   end if
11: end for
```

Binary exponentiation algorithms are used for modular exponentiations in RSA and for Elliptic Curve Scalar Multiplications (ECSMs) on additive group structures. The double-and-add-always algorithm (ECC), the square-and-multiply-always algorithm (RSA) as well as the Montgomery ladder algorithm (RSA and ECC) are examples for algorithms of this kind. They are a perfect target for side-channel attacks based on *localized EM analysis* because the use of registers depends on the processed secret while the processing sequence has a constant timing. Binary exponentiation algorithms typically consist of a main loop and process one secret bit in each loop iteration. Secure implementations of cryptographic algorithms typically also contain uniform operation sequences in each loop iteration, which are independent of the secret, as a countermeasure against simple side-channel analyses. However, the operations are performed on a different set of registers depending on the value of the currently processed secret bit. An attacker can use localized EM analysis to detect the usage sequence of those registers to derive the secret.

Algorithm 1 presents an abstract example showing the relevant properties of such algorithms. The depicted pseudo-algorithm has an uniform operation sequence and contains two operations in the loop iteration which are prone to location-dependent leakage because the register usage depends on the currently processed secret bit. In Lines 3 and 7, either register a or b are read. In Lines 5 and 9, a result is either written to register a or b. As described in Sect. 2, the registers are placed on an integrated circuit and an attacker can use location-dependent leakage to detect which of two registers is used in every iteration to recover the secret. The usage of the register c is independent of the secret bit and therefore not relevant for an attacker.

Figure 2 depicts a simplified example of a recorded EM trace vector $\mathbf{t} = (t_1, \ldots, t_T)$ containing the loop part of a binary exponentiation algorithm.

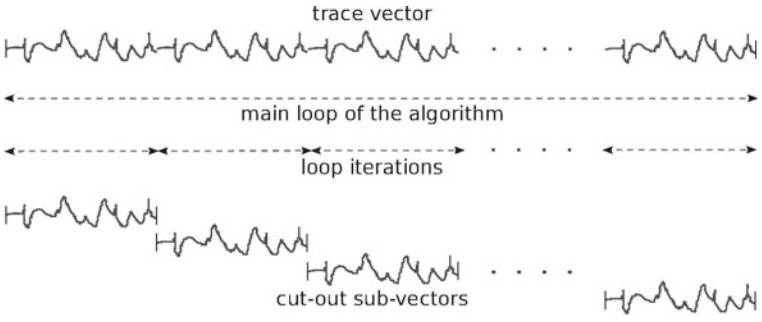


Fig. 2. Segmentation of trace vector $\mathbf{t}$ into sub-vectors $\mathbf{t_i}$

This trace vector is split into sub-vectors each containing one loop iteration, $\mathbf{t_i} = (t_{(1+(i-1)\frac{T}{D})}, \ldots, t_{(i\frac{T}{D})})$, $1 \leq i \leq D$ and D the number of loop iterations. The segmentation is derived from visual inspection and can be enhanced through cross-correlation of sub-vectors with the trace. All sub-vectors $\mathbf{t_i}$ are samples of the same operation sequence while processing a different secret bit d_i. Thus, the values from the different sub-vectors can be seen as belonging to same points in time within the loop iterations.

The measured power consumption of digital hardware processing uniformly distributed data consists of operation- and data-dependent parts and electronic noise and is typically distributed according to a Gaussian function. At certain relative points in time in the sub-vector set, two different registers are used in Alg. 1 which are placed at different locations. Hence, power is consumed at different locations on the circuit. Each location leads to a Gaussian distribution of the consumption. The goal of an attacker is to distinguish the distributions and to partition the sub-vectors $\mathbf{t_i}$ into two sub-sets. One set contains the sub-vectors where one register was used, the other one the sub-vectors where the other was used. This equals recovering the secret. In order to do this, an attacker must find similarities among the sub-vectors $\mathbf{t_i}$ which are due to the location-dependence. The following side-channel attacks can be applied to exploit such location-dependent leakage.

Side-Channel-Based Collision Attacks. Collision attacks [20,19] can be used to find similar sub-vectors $\mathbf{t_i}$ and classify them into two groups, hence, recovering the secret. Collisions can for instance be detected through least-square tests [19] or correlation [13,22]. A collision attack can be performed using a single recorded trace, where the secret exponent or scalar is processed. Data-dependent influence of the power consumption might make correct classifications more difficult. If the cryptographic protocol allows multiple executions with a constant secret and different processed data, those can be averaged to reduce this.

Template Attacks. Template attacks require a profiling phase using a known exponent or scalar. Fortunately, when exploiting location-based information, a public exponentiation can be used for profiling even if it uses a different base (e.g.,

base point in the ECSM). This is different to exploiting data-dependent leakage where templates are built to characterize certain intermediate data values. Such template attacks on binary exponentiation algorithms [2,12] require profiling with a chosen exponent or scalar and multiple templates. In case of exploiting location-based information, only two templates are required to classify all sub-vectors $\mathbf{t_i}$ into two groups. The sub-vectors from a profiling trace are grouped according to the bit values of the known exponent and the two groups' mean vectors and covariance matrices are used as templates. Since different data is processed in each sub-vector, data-dependencies are reduced in this process. In the attack, the sub-vectors $\mathbf{t_i}$ of a recorded trace with a secret exponent or scalar are matched against the templates to recover the scalar.

Finding Location-Dependent Leakage. If the attacker can observe executions with known data, he can employ a difference-of-means test to find eligible locations. Generally, the sample values from a *localized EM analysis* are distributed according to a Gaussian mixture of two superposed normal distributions instead of one because power is consumed at different locations when location-based information is leaked. This property can be used in order to find eligible locations on the surface of an integrated circuit die where an attack can be performed. Such Gaussian mixtures can be estimated using the expectation-maximization algorithm. Alternatively, a chi-square test or simply search of high variances can be performed.

4 ECC Case Study - A Proof-of-Concept

As a proof-of-concept, we employed a reduced template attack [11] to exploit location-dependent leakage of an FPGA implementation of the Elliptic Curve Scalar Multiplication (ECSM).

4.1 ECSM Implementation

The FPGA-based hardware implementation of the ECSM takes affine x- and y-coordinates of the base point x_P and the scalar d as input and returns affine x- and y-coordinates of the resulting point $d \cdot P$. The design uses an elliptic curve defined over the binary field $GF(2^{163})$ with elements represented in polynomial base. The field polynomial, curve parameters a and b, base point (x_P, y_P) and base point order n are published by NIST [14] under the denominator *Curve B-163*. The Montgomery ladder ECSM algorithm presented by López and Dahab [10] with projective coordinates during the Montgomery ladder and no repeated computation of an inverse in $GF(2^m)$ is employed and depicted in Alg. 2. The affine x- and y-coordinates are computed from the resulting point's projective coordinates in a routine denoted by *Mxy* from López and Dahab [10].Different to the description of López and Dahab [10], we use ∞ and P instead of P and $2 \cdot P$ as starting points for the Montgomery ladder in order to permit zero-valued most significant bits of the scalar.

Algorithm 2. Montgomery ladder ECSM algorithm.

Input: Scalar $d = d_D d_{D-1} ... d_2 d_1$ with $d_i \in \{0, 1\}$, Point $P = (x_P, y_P) \in E$, Curve Parameter b
Output: Point $Q = d \cdot P = (x_Q, y_Q)$

1: $X_0 \leftarrow 1$, $Z_0 \leftarrow 0$, $X_1 \leftarrow x_P$, $Z_1 \leftarrow 1$
2: **for** $i = D$ downto 1 **do**
3: $\quad T \leftarrow Z_{1-d_i}$
4: $\quad Z_{1-d_i} \leftarrow (X_{1-d_i} \cdot Z_{d_i} + X_{d_i} \cdot Z_{1-d_i})^2$
5: $\quad X_{1-d_i} \leftarrow x_P \cdot Z_{1-d_i} + X_{1-d_i} \cdot X_{d_i} \cdot T \cdot Z_{d_i}$
6: $\quad T \leftarrow X_{d_i}$
7: $\quad X_{d_i} \leftarrow X_{d_i}^4 + b \cdot Z_{d_i}^4$
8: $\quad Z_{d_i} \leftarrow T^2 \cdot Z_{d_i}^2$
9: **end for**
10: $(x_Q, y_Q) \leftarrow Mxy(X_0, Z_0, X_1, Z_1, x_P, y_P)$
11: **return** (x_Q, y_Q)

The algorithm processes the 163 bit scalar d bitwise employing a uniform operation sequence to prevent simple side-channel analyses and C-safe fault attacks [4]. To protect the Montgomery ladder against DPA, the projective coordinates of the input point are randomized [3]. The registers T, x_P and b are used equally, independent of the scalar bit values. The working registers X_0, Z_0 and X_1, Z_1 are used differently, depending on the scalar bits d_i. This makes the implementation prone to localized EM analysis. Those registers' design is completely equal on the RTL level. No further manual effort or constraints were employed during FPGA synthesis and placement on the die.

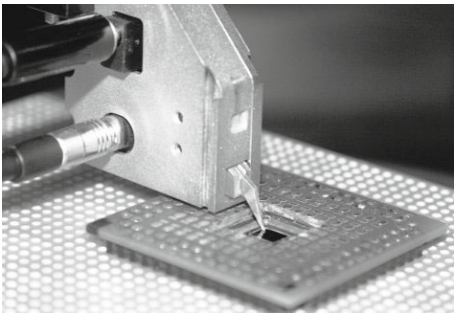


Fig. 3. Near-field probe close to the back-side-decapsulated surface of the die

4.2 Measurement Setup

Our measurement setup is depicted in Fig. 3. We decapsulated a *Xilinx Spartan-3 (XC3S200)* FPGA from the back-side as suggested by Skorobogatov et al. [21]. Back-side decapsulation is less complex than front-side decapsulation for smartcards and many plastic packages. An inductive near-field EM probe with

a $100\,\mu m$ resolution and a $30\,dB$ amplifier was used at a close distance to the surface of the die. The near-field probe was moved over the surface by an x-y-table with a positioning accuracy of $50\,\mu m$. At every location, one trace was recorded at a sampling rate of $5\,GS/s$ and compressed to one sample per clock cycle, extracting the difference of the maximum peak to the minimum peak EM value in every cycle. This was done to reduce the amount of data.

4.3 Template Attack

In our case study, we assumed that an attacker can observe public operations using a public and known exponent on the device he is attacking. We used these to find eligible locations for the attack and to build templates. In ECC, the public operation can for instance be a signature verification using a known scalar for the ECSM.

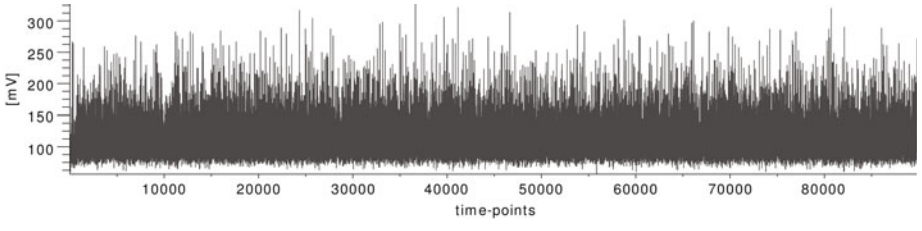
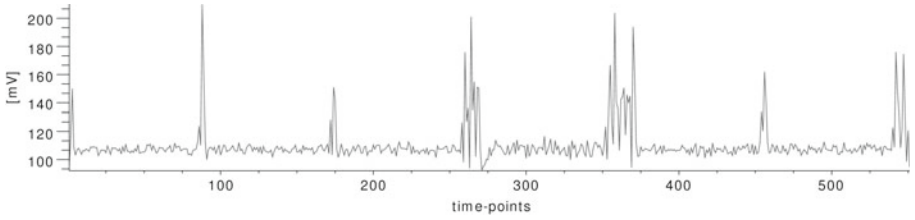


Fig. 4. Recorded EM trace $\mathbf{t}$ at location $(x, y) = (37, 42)$ as an example

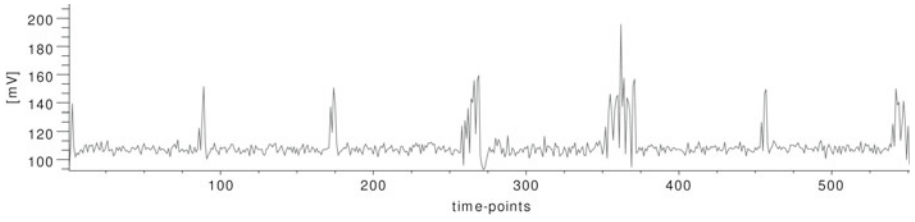
Profiling. Figure 4 depicts an EM trace vector as an example which was recorded using our measurement setup and a known scalar. It includes about $90\,k$ points in time, covering the main loop of Alg. 2 (Lines 3 to 8). The recorded trace vector $\mathbf{t}$ was split into 163 sub-vectors $\mathbf{t_i}$ corresponding to 163 scalar bits as described in Sect. 3. All sub-vectors correspond to an uniform operation sequence which is depicted in Alg. 2, Lines 3 to 8 and include about 550 EM values, thus, cycles each. The addressing of the registers depends on the value of the corresponding scalar bit which will be exploited during the attack.

The sub-vectors were assigned to two sets according to the known scalar bits. Figure 5(a) and Fig. 5(b) depict the mean vectors $\mathbf{m_0}$ and $\mathbf{m_1}$ of those two sets. Figure 5(c) shows the difference-of-means as a black graph. The Figure also depicts a grey zero line including a confidence interval at a confidence level of 99.9 % marked by two grey graphs. Points in time, where the black difference-of-means graph exceeds the zero-region, which is confined through the grey graphs, are clearly visible, showing that this design leaks location-dependent information. In our case study, we used the mean vectors $\mathbf{m_0}$ and $\mathbf{m_1}$ as reduced templates without covariance matrices and employed a least-square matching.

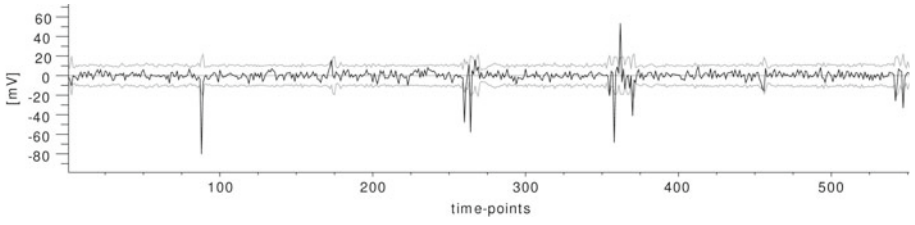
We performed the described difference-of-means test on every location on the surface of the die using a single recorded trace at every location. Figure 6 presents an (x, y) map of the greatest *absolute* difference-of-means in mV for each location



(a) Estimated mean $\mathbf{m_0}$ of sub-set where d_i is zero.



(b) Estimated mean $\mathbf{m_1}$ of sub-set where d_i is one.



(c) Difference-of-means $\mathbf{m_0} - \mathbf{m_1}$ as a black graph including a confidence interval depicted by two grey graphs around a grey zero line.

Fig. 5. Sub-vector means and difference of means

on the die. The maximum of about 63 mV is clearly significant compared to the amplitudes from Fig. 4. *A pattern of locations with high information leakage* can be observed. For the actual attack, the location with the greatest difference-of-means was used.

Figure 7 depicts a map with average EM amplitudes of the recorded traces. It can be observed that regions with *high EM amplitudes are not congruent to regions leaking most location-based information.*

Significance of the Location Dependence. In order to illustrate the significance of the location dependence, we narrowed the previously described analysis down to a single point-in-time. We chose the time-point 88 within the trace sub-vectors which exhibits a significant information leakage in Fig. 5(c). Figure 8 shows a map where each value corresponds to the *signed* difference-of-means when looking at this single operation. Most of the map is colored in turquoise indicating a 0 mV difference-of-means, thus, no information leakage. However, while the probe is moved over the die, it gets closer to cells belonging to one of two registers. In those regions, which are colored in blue to pink, this first

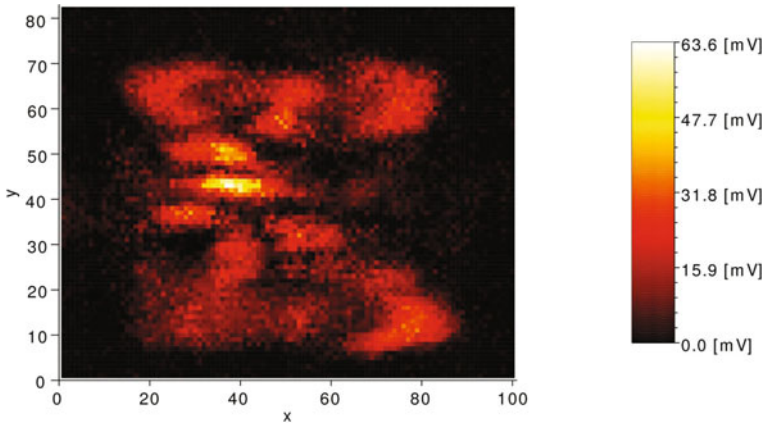


Fig. 6. Greatest *absolute* difference-of-means for all locations

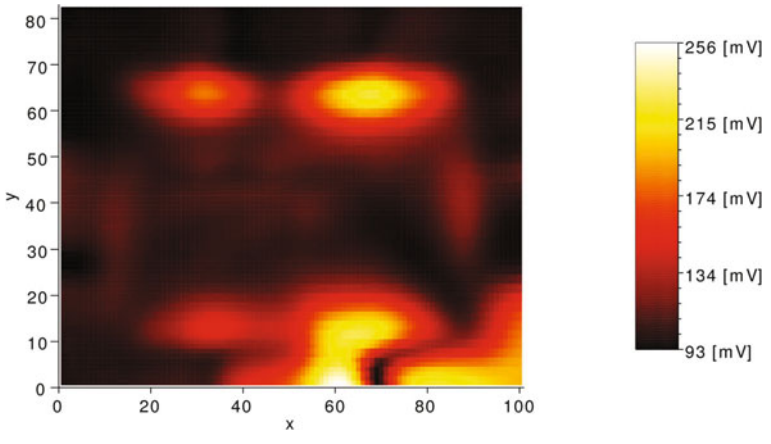


Fig. 7. Average EM amplitude for all locations

register leads to higher EM values than the other one. At other regions, colored in green to red, the probe gets closer to cells belonging to the other register. In those regions, the other register leads to higher EM values and, therefore, the difference-of-means changes sign. This shows that there are significant location-dependences of the information leakage. The oppositely signed differences will likely cancel out each other when analyzing the power consumption of the entire device.

Generally, it is hard to balance the physical implementation of registers to achieve identical power consumptions. The differences in the registers' power consumption might be detectable in overall power consumption measurements.

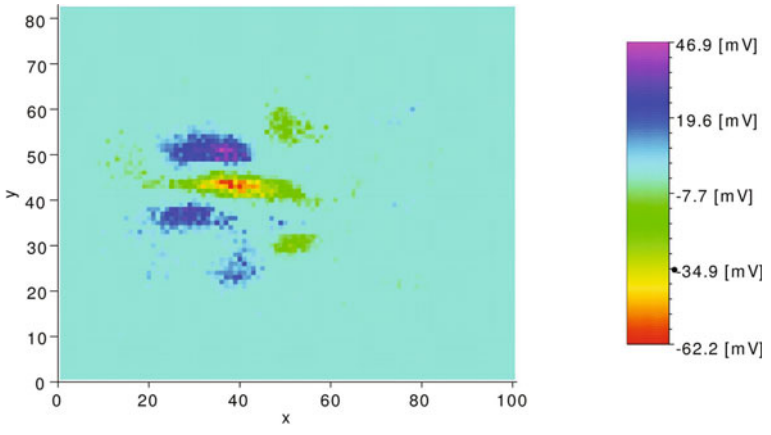


Fig. 8. *Signed* difference-of-means for point-in-time 88 at all locations

However, those differences are usually small and hard to exploit in single trace attacks. The additional localized aspect provides a significant improvement in this respect.

Attack. A single trace is used by the attacker to recover the secret scalar. Retrieving the scalar is equivalent to a total break of the system since the secret key can be computed easily from it. The segmented sub-vectors are compared to both templates using a least-square distance test. In this practical experiment we classified 161 of 163 secret scalar bits correctly, leaving 2 erroneous bits. This *proves that location-dependent leakage can be successfully exploited in practice.* The remaining erroneous bits are most likely due to electric and data-dependent noise. The error probability is highest for those recovered bits, where the classification was least decisive. In this manner, an attacker incrementally brute-forces bits where the difference between the two least-square values is smallest until the recovered secret is correct. In this practical experiment, we recovered the correct scalar after brute-forcing 14 bits at maximum. Hence, even if the classification does not provide a full recovery of the scalar, it *reduces the search space to a practical level.*

We tested recording multiple traces with a constant scalar and different input data to reduce the data-dependent influence through averaging. The success rate increased to 100 % using only 3 averaged traces.

5 Countermeasures

Designers can generally use traces with a known exponent or scalar to perform difference-of-means tests to look for leaks of location-dependent information. Countermeasures against power or EM analyses such as e.g., employing the Montgomery ladder, randomization of projective coordinates [3], base point

Algorithm 3. Our countermeasure for Alg. 2.

9: $r \leftarrow random \in [0, 1]$
10: $c \leftarrow swap\_state \oplus r$
11: $T \leftarrow X_0 + X_1$
12: $X_0 \leftarrow T - X_{1-c}$, $X_1 \leftarrow T - X_c$ $\{swap\ X_0\ and\ X_1\ if\ c = 1\}$
13: $T \leftarrow Z_0 + Z_1$
14: $Z_0 \leftarrow T - Z_{1-c}$, $Z_1 \leftarrow T - Z_c$ $\{swap\ Z_0\ and\ Z_1\ if\ c = 1\}$
15: $swap\_state \leftarrow r$

blinding or exponent blinding *do not prevent location-dependent information leakage*. Exponent blinding only prevents template attacks based on location-dependent leakage if it is also employed for the public operation. However, this introduces a significant computational overhead and does for example not prevent collision attacks based on location-dependent leakage. *Randomizing the assignment of registers to physical locations* on an integrated circuit *generally prevents location-dependent information leakage because the relation between location-based information and the secret is eliminated.*

Accordingly, we present a countermeasure for implementations of the López-Dahab Montgomery ECSM which is depicted in Alg. 3. The additional operations are integrated into the ECSM Alg. 2 within the loop beyond Line 8 and swap the register contents of X_0, X_1 and Z_0, Z_1 according to a chosen random number r. The value of the scalar bit d_i, thus, the addressing of the registers within the loop, is inverted according to whether the registers are swapped ($swap\_state$). The countermeasure is uniform in its operation sequence, hence, not detectable through simple analyses. It requires 2 field additions and 4 field subtractions (equal to addition in $GF(2^m)$) in every loop iteration which resulted in a computational overhead of about 4 %.

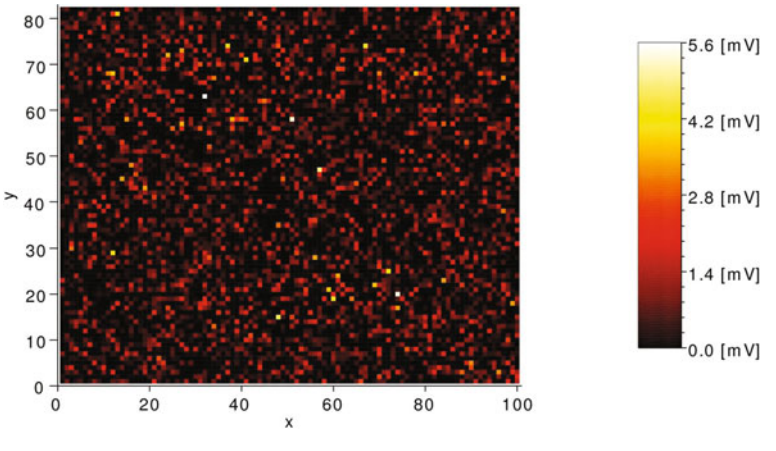


Fig. 9. Greatest absolute difference-of-means when using our countermeasure

We repeated the known scalar difference-of-means test. Figure 9 depicts a map of the resulting greatest absolute difference-of-mean values. A comparison to the results in Fig. 6 from the unprotected implementation confirms that the random swapping of register locations effectively destroys the relation between location-based information and the secret.

6 Conclusion

We present how location-dependent leakage instead of data-dependent leakage of cryptographic implementations can be used for side-channel attacks. As a conclusion we promote that in all implementations of affected cryptographic algorithms, the assignment of concerned registers to physical locations should be repeatedly randomized by swapping their locations at random times during execution. This randomization also renders it unnecessary to carefully balance the physical implementation of those registers.

Acknowledgements. The work presented in this contribution was supported by the German Federal Ministry of Education and Research in the project *RESIST* through grant number 01IS10027A.

References

1. Agrawal, D., Archambeault, B., Rao, J., Rohatgi, P.: The EM Side–channel(s). In: Kaliski Jr., B.S., Koç, C., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, C., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
3. Coron, J.S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, C., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
4. Fan, J., Guo, X., De Mulder, E., Schaumont, P., Preneel, B., Verbauwhede, I.: State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In: IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2010 (2010)
5. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In: Koç, C., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
6. Hofreiter, P., Laackmann, P.: Electromagnetic espionage from smart cards - attacks and countermeasures. Secure 6, 40–43 (2002)
7. Kirschbaum, M., Schmidt, J.M.: Learning from electromagnetic emanations - a case study for iMDPL. In: Workshop Proceedings COSADE 2011, pp. 50–55 (2011)
8. 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)
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

10. López, J., Dahab, R.: Fast Multiplication on Elliptic Curves over GF(2m) without Precomputation. In: Koç, C., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 316–327. Springer, Heidelberg (1999)
11. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Advances in Information Security. Springer-Verlag New York, Inc., Secaucus (2007)
12. Medwed, M., Oswald, M.E.: Template Attacks on ECDSA. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 14–27. Springer, Heidelberg (2009)
13. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
14. National Institute of Standards and Technology: Recommended elliptic curves for federal government use (July 1999)
15. Quisquater, J.J., Samyde, D.: Electromagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
16. Real, D., Valette, F., Drissi, M.: Enhancing correlation electromagnetic attack using planar near-field cartography. In: Design, Automation Test in Europe Conference Exhibition, DATE 2009, pp. 628–633 (April 2009)
17. Sauvage, L., Guilley, S., Flament, F., Danger, J., Mathieu, Y.: Cross-correlation cartography. In: International Conference on Reconfigurable Computing and FPGAs (ReConFig 2010), pp. 268–273 (December 2010)
18. Sauvage, L., Guilley, S., Mathieu, Y.: Electromagnetic radiations of fpgas: High spatial resolution cartography and attack on a cryptographic module. ACM Trans. Reconfigurable Technol. Syst. 2, 4:1–4:24 (2009)
19. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
20. Schramm, K., Wollinger, T., Paar, C.: A New Class of Collision Attacks and its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
21. Skorobogatov, S.: Optical fault masking attacks. In: 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 23–29 (August 2010)
22. Witteman, M., van Woudenberg, J., Menarini, F.: Defeating RSA Multiply-always and Message Blinding Countermeasures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 77–88. Springer, Heidelberg (2011)

Towards Different Flavors of Combined Side Channel Attacks

Youssef Souissi[1], Shivam Bhasin[1], Sylvain Guilley[1], Maxime Nassar[1,2], and Jean-Luc Danger[1]

[1] TELECOM ParisTech, 46 rue Barrault, 75634 Paris, France
[2] Bull TrustWay, 78340 Les Clayes-sous-Bois, France
firstname.lastname@TELECOM-ParisTech.fr

Abstract. Side Channel Attacks (SCA) have come a long way since first introduced. Extensive research has improved various aspects of SCA like acquisition techniques, processing of traces, choice of leakage model, choice of distinguishers etc. As a result, side-channel countermeasures have also improved. It is difficult to defeat such countermeasures and requires a huge number of traces. So far, only a few works studied the combination of SCA. In this paper, we put forward two methods to combine different attacks to accelerate SCA or to reduce the number of traces to attack. The first method is a combination of commonly used distinguishers. We provide a theoretical method and an empirical approach to combine Pearson and Spearman correlation coefficients. The second method suggests a combination of different measurements corresponding to the same activity. A metric to assess this combination using information theory is also given. Both methods are supported by application on real traces. The gain is expressed in terms of reduction in number of traces to attack. We report a gain of 50% for the first method and 45% for the second method.

Keywords: Correlation Power Analysis (CPA), Spearman Correlation Analysis, Gini Correlation, Combined Side Channel Attacks.

1 Introduction

Side Channel Attacks (SCA) have become an important issue in applied cryptography. SCA pose a serious practical threat to physical implementation of secure devices. These attacks exploit unintentional physical leakage, such as the timing information, power consumption or radiated magnetic field. Since Kocher et al. [15] introduced Differential Power Analysis (DPA) in 1998, an intensive research has been done to improve and extend side-channel attacks.

Generally speaking, countermeasures against SCA try to increase the minimal number of traces to attack. These countermeasures rely on adding noise to the system, masking or hiding the sensitive data. Another countermeasure proposes to change the secret key used for encryption and decryption regularly [14] which is often used in real-life devices. Such countermeasures make the number of traces

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 245–259, 2012.

to attack a scarce resource. Hence, there is a need to find methods to accelerate these attack.

Some previous works compare side-channel distinguishers.[1] A recent study shows that most univariate distinguishers are equivalent asymptotically [16], and that they only differ by statistical artifacts that are data-dependent when the environmental noise tends to zero. However, very few papers have tried to combine these distinguishers or methods to improve existing attacks.

In [22], a combination of timing and power attack is used to attack RSA. Two different kinds of combined attacks were put forward in [1]. In the first attack, a set of traces is partitioned using two different leakage models: a 4-bit model and a mono-bit model. A third model built combining these two models is used for the attack. In the second method, some relevant time samples are localised and combined. The authors of [1] suggest to compute the product of the correlation coefficient at relevant time samples. Both these methods result in a faster convergence towards a success rate of 100%. Another related work is [24], where the authors tend to concatenate and combine electromagnetic (EM) and power traces. The problem of the method proposed in [24] is that principle component analysis (PCA) when applied to the traces without normalization, favors the one with higher variance. Some works also tried to combine SCA with faults as in [3,23] to accelerate the attack and open new attack paths.

In this article, we put forward two new methodologies to combine common side channel attacks in order to accelerate the key recovery. The combination can be anticipated at different stages varying from the acquisition to the attack. The two combinations we propose are:

Combination of Distinguishers. The choice of a proper side-channel distinguisher is essential for a successful SCA. We propose to combine different distinguishers to accelerate the attack. We demonstrate our methodology by combining the Pearson and the Spearman correlation coefficients, though more than two distinguishers can also be combined. We propose two methods for combination. Complex correlation coefficients combining the advantages of different distinguishers is popular research interest in the field of statistics. Such correlation can be seen as a theoretical combination. We show that the Gini correlation, a combination of the Pearson and the Spearman correlation is an optimal distinguisher for SCA. Some practical methods can also be applied to use the results of different distinguishers increasing the signal to noise ratio. Under some conditions, we show that the combination of these coefficients is possible and leads to a more powerful SCA.

Combination of Measurements. A common practice to carry out Electromagnetic Analysis (EMA [20]) is to acquire the strongest and most obvious leakage points on the device. However, there are other points which also leak exploitable information. We propose to acquire multiple simultaneous leakages

[1] A distinguisher is basically a statistical test, that aims at putting forward any bias. Some examples include a difference of means [15], a covariance [11], a correlation (linear [6] or rank-based [5]), mutual information [9] or variance [25].

from different leakage points using multiple antennae. These multiple leakages for a single activity could be combined for an efficient SCA. Multi-channel attacks have already been introduced in [2] for mono-bit DPA and template attacks. In this article we give a more generic outlook towards combining measurements using any distinguisher. We also provide a metric based on information theory to test if the possibility of combination exists for a given pair of traces collections.

The rest of this paper is organised as follows. Section 2 gives a general background on power analysis. Sections 3 and 4 detail the aforementioned combination attacks along with corresponding results on real traces. Finally, Section 5 lists the conclusions drawn and possible extensions to this work.

2 Power Analysis: General Background

2.1 The Principle

Power analysis consists of exploiting dependencies between the manipulated data and the analog quantities (power consumption, electromagnetic radiation ...) leaked from a CMOS circuit. In practice, it is difficult to model the signal leaked by a hardware implementation. The reason is that hardware implementations manipulate a large amount of data in parallel, but we target only a few bits of this data when performing power analysis. Suppose that D power consumption traces are recorded while a cryptographic device is performing an encryption or a decryption operation. The attacker chooses an intermediate result of the cryptographic algorithm. The intermediate value can be modelled as a deterministic function that takes two parameters: a known d which can be either the plain text or the cipher text and a secret (unknown) sk. Indeed, sk is a small part of the cryptographic key and can take K possible values referred to as key hypotheses, denoted by $\breve{k}$. In what follows, we denote the intermediate value by $v_{d,sk}$ generating a physical leakage, $l_{d,sk}$. In the literature of SCA, the leakage $l_{d,sk}$ is assumed to be composed of two terms: a deterministic term, $\phi(v_{d,sk})$, and an independent noise term ϵ_d. With these notations, the actual leakage $l_{d,sk}$ is written as follows:

$$l_{d,sk} = \phi(v_{d,sk}) + \epsilon_d. \tag{1}$$

Practically, a leakage model[2] is based on a logical function which enables an attacker to compute a hypothetical intermediate value $h_{d,\breve{k}} = h_{func}(v_{d,\breve{k}})$ for every possible $\breve{k}$ key hypothesis and d. This way, the leakage measurements are implicitly classified into several partitions, according to the hypothetical intermediate values computed for each key hypothesis. Eventually, the attacker uses a statistical test, referred to as *distinguisher* $\Delta_{\breve{k}}$, to compare $h_{d,\breve{k}}$ with $l_{d,k}$. Formally, the attacker builds a score vector $\Delta_{vect} = (\Delta_{\breve{k}})_{\breve{k}=1}^{K}$. The key candidate $\breve{k}$ that is the most likely to be the right key hypothesis (*i.e* the secret key sk) is the

[2] The most commonly used power models for characterizing the power consumption are the Hamming distance (HD) and the Hamming weight (HW) [6].

one which corresponds to the absolute maximum score $\breve{k} = \arg\max_k |\Delta\breve{k}|$. The distinguisher used by the Correlation Power Analysis (CPA [6]) is the Pearson's product moment correlation coefficient ρ, defined as:

$$\rho_{X,Y} = \frac{\mathbb{E}[(X - \mu_X) \cdot (Y - \mu_Y)]}{\sigma_X \cdot \sigma_Y} = \frac{Cov(X,Y)}{\sigma_X \cdot \sigma_Y} \quad , \tag{2}$$

where X and Y are two random variables with expected values μ_X and μ_Y and standard deviations σ_X and σ_Y, respectively, $\mathbb{E}$ is the expected value operator and *Cov(X,Y)* is the Covariance between X and Y. ρ is a dimensionless index and is invariant to affine transformations of either variable.

3 Combination of Distinguishers

Gini correlation, which is often used in computing finance and income distribution, combines the advantages of the Pearson correlation ρ and the Spearman correlation r. In this section, we first introduce the Gini correlation coefficient followed by its application to side-channel analysis.

3.1 Mathematical Background

Let X be a random variable that takes its values from a finite set $\mathcal{X}$ (*e.g.*, $\mathcal{X} = \mathbb{F}_p^q$). We denote by x a particular element from $\mathcal{X}$. The probability density function (*pdf*) of the event $(X = x)$ is referred to as $p_X(x)$. Suppose we want to best approximate Y with another variable X based only on the knowledge of their joint distribution $P_{X,Y}(x, y)$. The problem is to find a function $\phi(.)$ of X that best fits Y among all possible forms of ϕ. In our study, the variable X is deterministic since it is theoretically predicted from a known cryptographic process. Whereas, the variable Y is a real measure acquired by an oscilloscope. Thus, for sake of clarity, the variable X is called *the prediction* and Y *the measurement*. Depending on the causal connections between X and Y, their true relationship may be linear or non linear. The independence of X and Y implies that they are uncorrelated. The converse is true only under the Gaussian assumption. In fact, this assumption states that the joint distribution of X and Y is bivariate normal. In this case, X and Y are said to be *jointly Gaussian* variables.

However, regardless of the true nature of the relation, a linear model can be used for an initial approximation when X and Y are scalar:

$$Y = \phi(X) + \epsilon = (\alpha + \beta X) + \epsilon, \tag{3}$$

where α is the *intercept*, β is the *slope* of the line, and ϵ is the error of the approximation.

Definition: Jointly Gaussian
A set of n random variables $X_1, X_2, \ldots, X_n$ are jointly Gaussian if $\sum_{i=1}^{n}(a_i X_i)$ is a Gaussian random variable $\forall$ real a_i, with $i \in [1..n]$.

A common pitfall about the validity of the Gaussian assumption is to check only that X and Y are drawn from normal distributions. If X and Y are each individually Gaussian then this does not imply that they are jointly Gaussian. Generally, a joint distribution $P_{X,Y}(x, y)$ is said to be bivariate normal if the four conditions **normal conditional distribution**, **linearity**, **homoscedasticity** and **normal marginal distribution** are satisfied [4]. Homoscedasticity means that the conditional distribution of Y given $X = x$ has finite variance for each x. Moreover, under these conditions, ϵ must be drawn from a zero mean normal distribution. In other words, ϵ is a random variable strictly independent from X and a linear function ϕ characterizes the dependence between X and Y, entirely. Estimation theory shows that under the Gaussian assumption, ρ is the best tool to totally characterize the association (purely linear) between X and Y [27,7,13]. However, in real situations, it is hard to get a perfect binormal joint distribution. In such situations, the higher the deviation from the Gaussian assumption is, the lower the efficiency of ρ is. In this case, other correlation coefficients have been developed to be more robust[3] than the Pearson correlation. Some examples include the Spearman coefficient r and Kendall's tau r_τ (rank correlations), biserial and tetrachoric [17]. Spearman correlation measures both the linear and the non-linear relationship between the two variables, as it does not requires that the observations are drawn from a Gaussian distribution. It is a *non-parametric* coefficient that was first applied in side-channel context in [5].

In the literature of correlation analysis, there is no rule to determine whether the ρ will outperform its competitors or not, provided the deviation from Gaussianity is not excessive. In this insight, statisticians have recently started to investigate actual combinations between existing correlation coefficients, which bridge the gap between Pearson coefficient and its competitors.

3.2 Gini Correlation: A Mixture of Pearson and Spearman Coefficients

Pearson correlation, ρ, might perform poorly when the data is attenuated by non-linear transformations, in contrast to Spearman correlation, r. However, r is not as efficient as ρ under the Gaussianity. This robust alternative to ρ might lose its efficiency especially when the data involves different types of variables (*e.g.* discrete/continuous). Moreover, when the number of different values taken by either variables is small, then this might create a *problem of ties* (*i.e.* there is a tie while ranking the data. This affects considerably the quality of r [10]). In such cases, the loss of efficiency might not be compensated by the robustness in practice. For this purpose, statisticians have recently came with an interesting combination between Pearson and Spearman coefficients, namely Gini correlation (ξ), which has been proposed in [21].

[3] A statistical criterion that does not make any assumption about the joint distribution is said to be robust or distribution free.

Spearman correlation r , which is just ρ applied on already ranked data, can be defined using the notion of cumulative distributions as:

$$r_{(X,Y)} = \rho_{(F_X,F_Y)} = \frac{1}{\sigma_{F_X}\sigma_{F_Y}} Cov(F_X(X), F_Y(Y)) \tag{4}$$

where F_X and F_Y are the cumulative distribution of X and Y, respectively. Similarly to Eq (4) and Eq (5), the Gini correlation coefficient is given by:

$$\xi_{X,Y} = \frac{Cov(X, F_Y(Y))}{Cov(Y, F_X(X))} \tag{5}$$

Note that in general ξ is not symmetric i.e. $\xi_{X,Y} \neq \xi_{Y,X}$. In practice, the choice between the two forms, depends on the type of variables X and Y. In statistics, it has been reported that if, for instance, X is discrete and Y is continuous, then $\xi_{Y,X}$ would be a good choice.

Practical Computation of Gini Correlation and Properties. Consider n couples (X_i, Y_i) with $i \in [1..n]$ of independent variables drawn from a bivariate distribution. If these couples of variables are ordered (sorted from low values to high values) with respect to the X_i, new couples of variables $(X_{(i)}, Y_{[i]})$ can be generated, where $X_{(1)} < \cdots < X_{(n)}$. $Y_{[1]}, \cdots, Y_{[n]}$ are the related concomitants [18], which depend on the ordering of the X_i. As proposed in [21], $\xi_{Y,X}$ is computed as:

$$\xi_{Y,X} = \frac{\sum_{i=1}^{n}(2i - 1 - n)Y_{[i]}}{\sum_{i=1}^{n}(2i - 1 - n)Y_{(i)}} \ . \tag{6}$$

Note that, $\xi_{X,Y}$ is computed in the same way as $\xi_{Y,X}$, by just reversing the roles of X and Y. For more details about the Gini correlation coefficient, we refer the reader to [28].

We compared the three correlation functions (Pearson, Spearman and Gini) using simulated traces. The leakage function of a FPGA can be modelled as $\mathcal{L}(x) = \mathrm{HW}(x) + \alpha \cdot \delta(x)$, where $\delta(\cdot)$ is the Kronecker symbol. Here $\mathcal{L}$ is the leakage function and HW is the Hamming weight function. We verified this leakage model on public AES traces of DPA contest v2 [26] as shown in Fig. 1. We attack 8 bits at the output of an AES Sbox, using a linear base function of length 9 (8 bits and 1 constant). Thus, HW function can take 9 possible values (HW=0,1,2,3,4,5,6,7,8). The figure 2 (a) shows the comparison of three correlation coefficients as a function of α. A proper approximation for the Gaussian case is seen when α is 0 and all three correlation coefficients are equivalent empirically. When α is negative, Pearson correlation is not optimal. Spearman and Gini still perform very well as they are not sensitive to monotonic transformation. For positive α, Pearson is not suitable and Spearman also becomes less optimal as the function $\mathcal{L}$ is no more monotonic. Since the transformation is not drastic we see that the Spearman correlation tries to stabilise itself (Fig. 2 (b)). However, Gini does show some improvement over Pearson and Spearman. As stated earlier, Gini is a combination of Pearson and Spearman, we can say that combination can help in non-ideal cases. Next, we propose some empirical approach to combine Pearson and Spearman.

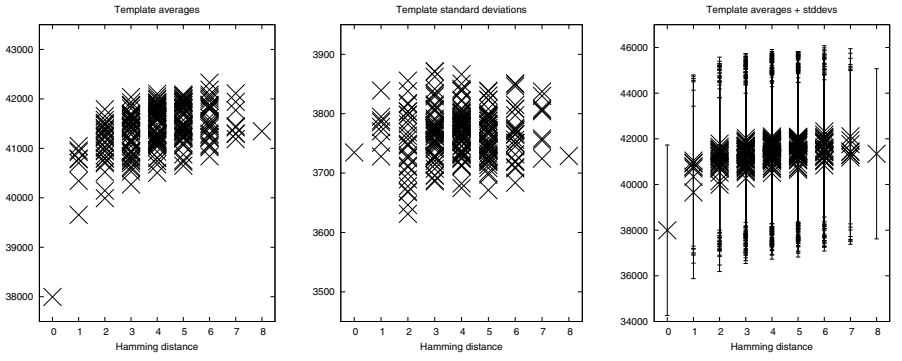
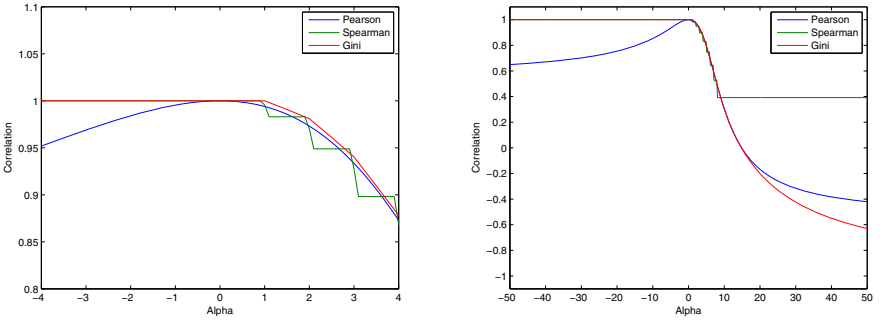


Fig. 1. Leakage function of Sbox 0 (DPA contest v2)



(a) (b)

Fig. 2. (a) Three correlation coefficients on the leakage function $\mathcal{L}$, extended in (b) to higher values of α

3.3 Pearson-Spearman Combination: An Empirical Approach

Why the Combination Works. As stated previously, most side channel attacks differ in the measurements partitioning process and the used distinguisher. Otherwise, they usually run iteratively and a new ranking of all secret hypotheses is created at each iteration. Our starting argument to combine two different distinguishers, $(\Delta_{\check{k}})_{sca'}$ and $(\Delta_{\check{k}})_{sca''}$, involves four observations: the first observation is that the two distinguishers are equivalent (*i.e.* similar evolution), in terms of *success rate* and *guessing entropy* security metrics [25], when performed in parallel, on the same set of side-channel traces. In addition, we observe that the *secret key* mostly keeps the same temporal position for both distinguishers unlike the false key hypotheses. We define the *predicted key*[4] as the key hypothesis that has the best rank PK for the current iteration. Its value is updated

[4] The *predicted key* is also known as *the best key*.

for each trace processed. We observed that the two distinguishers often do not have the same predicted key ($PK_{(\Delta_{\breve{k}})_{sca'}} \neq PK_{(\Delta_{\breve{k}})_{sca''}}$). But more importantly, this emphasizes the fact that $(\Delta_{\breve{k}})_{sca'}$ and $(\Delta_{\breve{k}})_{sca''}$ are statistically different, even if they are exploiting the same dependency. Eventually, the last observation is that the *secret key* is always ranked among the best ranked key hypotheses for both distinguishers. In fact, once the correct classification (partitioning) of the traces for each iteration is done, the attack succeeds as the *predicted key* is the actual *secret key*. The *secret key* achieves a Guessing entropy of zero when the attack succeeds. This is not the case for *false keys* which should have an unstable (random) rank.

Combination Formula. Consider two side-channel attacks, sca' and sca'', that verify the empirical observations mentioned before. Let $\Delta_{vect_{sca'}} = ((\Delta_{\breve{k}})_{sca'})_{\breve{k}=1}^{K}$ and $\Delta_{vect_{sca''}} = ((\Delta_{\breve{k}})_{sca''})_{\breve{k}=1}^{K}$, respectively. We can combine sca' and sca'' distinguishers by taking into account the scores given by both distinguishers for the same key hypothesis, $\breve{k}$. We use aggregate functions[5] Ψ for the combination (like the Max() and the Sum() functions). Similarly, Gini correlation can be imagined to use ratio as an aggregate function. For each key hypothesis, $\breve{k}$, a new score is generated by computing $\Psi((\Delta_{\breve{k}})_{sca'}, (\Delta_{\breve{k}})_{sca''})$, which is the aggregate function of $(\Delta_{\breve{k}})_{sca'}$ and $(\Delta_{\breve{k}})_{sca''}$. This way, a new vector of scores, denoted by $\Delta_{vect_{combi}}$ is built. An illustration of the combination mechanism is shown in Fig. 3.

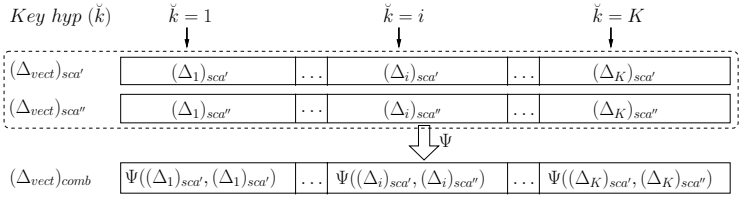


Fig. 3. The mechanism of combination using an aggregate function Ψ

3.4 Experimental Results and Discussion

Our measurement setup consists of one Altera Stratix-II FPGA soldered on an SASEBO-B platform, an 54855 Infiniium Agilent oscilloscope with a bandwidth of 6 GHz and a maximal sampling rate of 40 GSa/s, antennas of the HZ–15 kit from Rohde & Schwarz. We recorded 5000 side-channel traces (averaged 256 times) related to the activity of an unprotected DES crypto-processor.

The analysis of the marginal distributions of both the prediction X and the measurement Y, revealed that their joint distribution is not perfectly Gaussian. In fact, there is some deviation from the bivariate normal assumption; and

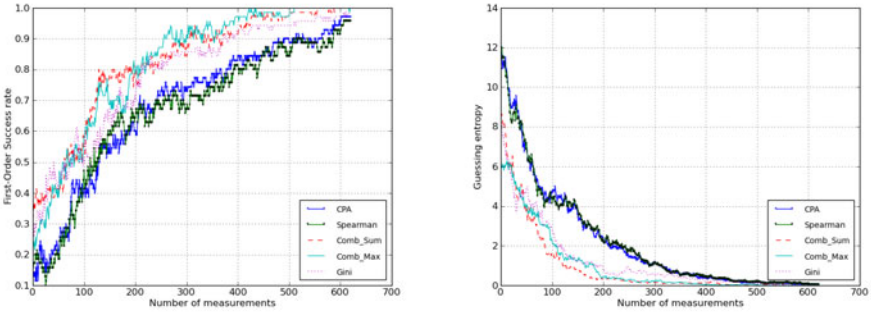
[5] An aggregate function is a special type of operator that returns a single value based on multiple rows of data.

therefore the Pearson correlation coefficient ρ might not be optimal. Moreover, Spearman correlation coefficient r might not be optimal too, because X takes a small number of different values which does not allow a reliable approximation by a normal distribution. As stated before, this might create a problem of ties, which affects considerably the quality of r. The experiment that we have conducted involves five side-channel attacks evaluated in term of their first-order success rate (SR) and Guessing entropy (GE) security metrics: correlation power attack (CPA), Spearman rank correlation, Gini correlation, and two empirical combination attacks. In this experiment, two aggregate functions have been investigated: the Sum(), and the Max(). These two attacks are denoted by $Comb_{Sum}$ and $Comb_{Max}$, respectively.

According to Fig. 4 (a) and Fig. 4 (b), CPA and Spearman attacks have similar behaviours. This agrees with our empirical statements stated previously. Clearly, the combined attacks (Gini Correlation, $Comb_{Max}$ and $Comb_{Sum}$) outperform CPA and Spearman attacks. As a matter of fact, for a SR threshold fixed at 80%, the number of traces needed to succeed in the combined attack is around *200 traces*. CPA and Spearman attacks need much more traces to do so (*400 traces*) and thus the gain is about 50%. Unsurprisingly, the GE metric shows a superior efficiency for the combined attacks as the rank of the *secret key* converges more rapidly toward the best rank much faster than CPA and Spearman attacks. Besides, for both metrics, Gini correlation is slightly less efficient than the empirical combinations, $Comb_{Sum}$ and $Comb_{Max}$. Let S_1, S_2 be two inputs of aggregate function with respective noise of standard deviation σ_1, σ_2. The signal-to-noise ratio (SNR) of the $Comb_{Sum}$ is $(S1+S2)/\sqrt{{\sigma_1}^2 + {\sigma_2}^2}$. When S_1 and S_2 are equal, the SNR of combination using $Comb_{Sum}$ is increased by $\sqrt{2}$. Similarly, the increase in SNR when two distinguishers are combined using $Comb_{Max}$ can be computed. However, Gini Correlation is more generic and might be more suitable in other empirical circumstances.

4 Combination of Measurements

Cartography is often used to reconstruct a dynamic image of the device using a sensor. An attacker can use this dynamic image in identifying the areas where the information leakage is the most intense [20]. As a matter of fact, the electromagnetic radiations correlated to a given process are not necessarily produced at the exact location of the processing zone. The power lines or clock paths leak more information and therefore power supply and ground networks as well as the clock buffer trees are of special interest. Another interesting source of leakage are the decoupling capacitors which can leak radiated emanations about an internal process. An EMA starts with research of a relevant leakage point for capturing EM radiations. An attacker can perform a complete cartography of the chip or carefully choose a decoupling capacitor. When dealing with complex cryptographic circuits which are often bulky, several leakage points are identified. Some of these leakage points provide enough leakage to mount a successful attack, however, the speed of attack could vary. A common practice is to choose



(a) (b)

Fig. 4. CPA, Spearman vs Combination: (a) Success Rate and (b) Guessing Entropy

the point which could lead to the fastest attack. We put forward a methodology to combine leakages from several leakage points in order to accelerate the attack. We use multiple antennae to capture the radiation from different chosen points, during a single encryption, for a given message and a fixed secret key. A combination of power measurement and EM measurement can also be used.

4.1 Theoretical Background

Information gain of a single attribute X with respect to class C, also known as mutual information between X and C, measured in bits is:

$$Gain_c(X) = I(X;C) = \sum_x \sum_c P(x,c) \log \frac{P(x,c)}{P(x)P(c)} \ . \tag{7}$$

Equivalently:

$$I(X;C) = H(X) - H(X|C) \ . \tag{8}$$

Here $H(X)$ is the entropy of X and $H(X|C)$ is the conditional entropy of X given C. Information gain is a measure of the strength of a 2-way interaction between an attribute X and the class C. 3-way interactions were introduced as interaction gain [12] which is equivalent to mutual information of 3-variables. Interaction gain is also measured in bits, and can be understood as the difference between the actual decrease in entropy achieved by the joint attribute (X, Y) and the expected decrease in entropy with the assumption of independence between attributes X and Y. Interaction gain can be considered equivalent to multivariate mutual information [8]. To simplify the calculation of entropy we consider the distribution of X is Gaussian. In this case entropy can be calculated as a function of standard deviation σ_x of X as:

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i) = \log_2(\sigma_x \sqrt{(2\pi e)}) \ . \tag{9}$$

Estimating entropy using Gaussian parametric method might not be very accurate but it works well in practice [19]. Nevertheless, other methods of estimating entropy can be equally applied.

$$I(X;Y;Z) = I(X,Y;Z) - I(X;Z) - I(Y;Z)$$
$$I(X;Y;Z) = (D+F+G) - (F+G) - (D+G) = \textbf{-G} \tag{10}$$

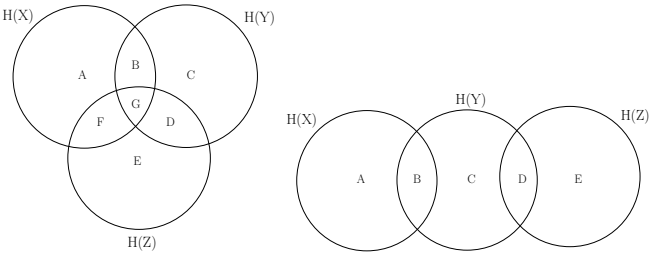


Fig. 5. Venn diagram representation of a case when combination is (a) possible, (b) not possible

The Venn diagram representation of interaction gain is shown in Fig. 5. As per Eq (10), interaction gain is equal to **-G**. If X and Y are independent, $I(X,Y;Z) = I(X;Z)+I(Y;Z)$. This means that the interaction gain $I(X;Y;Z)$ is zero. Interpreting from Fig. 5 (a) and (b) combination is possible when the information equal to D is added to $I(X;Z)$ with introduction of Y. This makes $I(X,Y;Z)$ = D+G+F. If D is zero, then the introduction of the Y is not providing any extra information.

To check this condition we propose a simple test. The possibility of combination (PC) can be calculated as a ratio:

$$\text{PC} = \frac{Max(I(X;Z), I(Y;Z))}{I(X,Y;Z)}\,. \tag{11}$$

For a combination to exist, the value of PC should lie between 0.5 and 1, where PC=1 will suggest no combination is possible. In the context of combined attacks, interaction gain can be directly applied. This is a profiling step because the knowledge of the secret key is required to calculate the value of PC. Alternatively, PC can also be used as a distinguisher. However, in this paper we show how to apply combination using CPA.

4.2 Practical Results

The experimental setup is the same as in Section 3.4. We target two decoupling capacitors on the backside of the FPGA which show emanations corresponding

to a DES execution. As the number of capacitors on the backside of the FPGA is small, we can use the trial-and-error method to choose the set of capacitance. A complex cryptographic operation generally consumes more than other operations on a chip which can be observed on an EM trace. We choose the capacitors where these cryptographic operation are obvious or clearly distinguishable on the EM trace. We collect two sets of 5000 traces from two chosen capacitors for the same data set. A crypto-processor is a bulky design and could be spread over different power banks in an FPGA which are terminated by different capacitors. Therefore, different capacitors leak more information about a certain part of the circuit. Here the partition can be seen as different Sboxes.

We start with testing the possibility of combination. Fig. 6 shows the PC values for the first Sbox. Fig. 6 (a) considers traces from two capacitance called C_1 and C_2 which are leaking relevant information. It can be seen that the value of PC is close to 0.5 when the value of mutual information is relevant. Fig. 6 (b) considers traces from a leaking capacitance and another point which is not leaking. Here the value of PC is close to 1. In other parts of the trace there is noise and the value of PC is changing randomly. Unfortunately, we did not have a set of traces for which the value of PC takes values between 0.5 and 1. It can be inferred from this experiment that combination is possible for the traces in Fig. 6 (a).

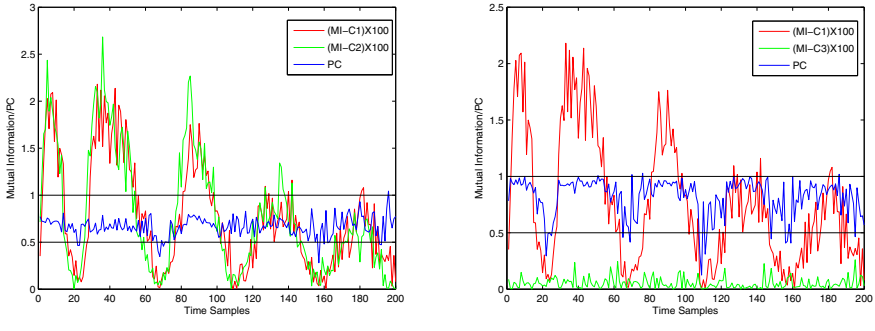


Fig. 6. Calculation of PC for two cases when combination is (a) possible, (b) not possible (mutual information of the two measurements is multiplied by 100 to visualize on the same scale as PC)

The next step is to observe practical application of combination of measurements using a common attack like CPA. We applied CPA on the traces collected from C_1 and C_2 independently. Table 1 summarises the result of CPA on each set of traces. These results are averaged over 30 attacks. We see that C_1 is better suited for Sbox no. $1, 2, 4$ and 8 and C_2 for the rest.

Before testing the combination, we concatenate traces of C_1 and C_2 together. Traces can be normalised before concatenation specially when techniques like

PCA are applied but if the traces are taken with the same scale then normalisation will not help a lot. In our experiments the traces are taken with the same scale on the oscilloscope therefore normalisation is not needed. We attack the concatenated trace by computing the Pearson correlation coefficient of the key hypothesis for each trace on each of the two sections of concatenated trace. To test the combination, we use an aggregate function Ψ as listed previously (Section 3.3). The aggregate function used in this experiment is the Sum() on the calculated coefficient values. The attack used to apply combination is CPA. Spearman, Gini and other coefficients can also be used. It is shown that Sum() can increase the SNR even if the two traces contain equivalent information. If the amount of information is not equivalent Sum() will further increase the SNR hence a faster attack. Performance of Sum() as a basis for combination has already been demonstrated in Fig. 4. We repeat the same attack using the Max() aggregate function. The results are slightly worse than for Sum().

The computation complexity of this attack is equivalent to processing a trace with twice the number of samples with minor overhead of applying the aggregate function. Two parallel attacks on non-concatenated traces will have similar computation overhead but concatenation makes it easy to manage the key hypotheses and apply aggregate functions.

Table 1 shows the number of traces to attack when combination is applied. We find that in each case the combination is better than individual attack and the gain varies from 4.16% to 44.86%. This also complies with our observations from the computed PC values. The values of PC computed previously and gain from table 1 cannot be directly compared because the basis of both the quantities are different. As mentioned before, some countermeasures change encryption key after a specific number of encryption to prevent SCA. Since the number of traces acquired is considered a scarce resource, we demonstrate that multiple measurements can be exploited for faster attack.

Table 1. No. of traces to attack using C_1, C_2 and combination of both.

Sbox No.	0	1	2	3	4	5	6	7
C_1	350	943	733	400	410	320	548	592
C_2	432	1073	720	980	176	281	551	192
$Comb\_sum$	212	750	397	251	165	270	448	184
Percent Gain	39.42	20.46	44.86	37.25	6.25	3.96	18.24	4.16

5 Conclusion

In this article we proposed two new methodologies of combined attacks. The first methodology combines commonly used side-channel distinguishers like Pearson and Spearman coefficient both theoretically (Gini correlation) and empirically (aggregate function). The second methodology combines measurements. We provide theoretical background based on information theory metrics to test combination by computing possibility of combination PC. Practical results show a gain

of about 50% with the first method and 45% with the second. We emphasize that we discuss methodologies to improve attacks in general, but there maybe cases which are better off using the prior techniques. Depending on the target, different distinguishers can be combined using appropriate aggregate functions. Choice of the aggregate functions depends on the practical behavior of distinguishers.

References

1. Aabid, M.A.E., Meynard, O., Guilley, S., Danger, J.L.: Combined Side-Channel Attacks. In: Chung, Y., Yung, M. (eds.) WISA 2010. LNCS, vol. 6513, pp. 175–190. Springer, Heidelberg (2011)
2. Agrawal, D., Rao, J.R., Rohatgi, P.: Multi-Channel Attacks. In: Walter, C.D., Koç, C., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 2–16. Springer, Heidelberg (2003)
3. Amiel, F., Villegas, K., Feix, B., Marcel, L.: Passive and Active Combined Attacks: Combining Fault Attacks and Side Channel Analysis. In: FDTC, September 10, pp. 92–102. IEEE Computer Society, Vienna (2007)
4. Arnold, B., Castillo, E., Sarabia, J.: Conditional specification of statistical models. Springer series in statistics. Springer, Heidelberg (1999)
5. Batina, L., Gierlichs, B., Lemke-Rust, K.: Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 341–354. Springer, Heidelberg (2008)
6. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
7. Dagnelie, P.: Statistique théorique et appliquée. Tome 2. Inférence statistique á une et á deux dimensions. De Boeck (2006)
8. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting Higher-Order DPA Attacks: In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 221–234. Springer, Heidelberg (2010)
9. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
10. Gravetter, F., Wallnau, L.: Essentials of statistics for the behavioral sciences. Thomson/Wadsworth (2008), http://books.google.com.nf/books?id=hcoYNW4BujYC
11. Guilley, S., Sauvage, L., Danger, J.L., Selmane, N., Pacalet, R.: Silicon-level solutions to counteract passive and active attacks. In: FDTC, 5th Workshop on Fault Detection and Tolerance in Cryptography, pp. 3–17. IEEE-CS, Washington DC, USA (2008)
12. Jakulin, A., Bratko, I.: Analyzing Attribute Dependencies. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 229–240. Springer, Heidelberg (2003)
13. Kamen, E.W., Su, J.: Introduction to optimal estimation. Advanced textbooks in control and signal processing. Control and Signal Processing Series. Springer, Heidelberg (1999)
14. Kocher, P.C.: Leak-resistant cryptographic indexed key update (March 25, 2003), United States Patent 6,539,092 filed at San Francisco, CA, USA (July 2, 1999)

15. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
16. Mangard, S., Oswald, E., Standaert, F.X.: One for All - All for One: Unifying Standard DPA Attacks. Cryptology ePrint Archive, Report 2009/449 (2009)
17. Myers, J., Well, A.: Research design and statistical analysis. L. Erlbaum Associates (1995)
18. Nagaraja, H.N.: Functions of concomitants of order statistics. Journal of the Indian Society for Probability and Statistics 7, 15–32 (2003)
19. Prouff, E., Rivain, M.: Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 499–518. Springer, Heidelberg (2009)
20. Sauvage, L., Guilley, S., Mathieu, Y.: ElectroMagnetic Radiations of FPGAs: High Spatial Resolution Cartography and Attack of a Cryptographic Module. ACM Trans. Reconfigurable Technol. Syst. 2(1), 1–24 (2009), http://hal.archives-ouvertes.fr/hal-00319164/en/
21. Schechtman, E., Yitzhaki, S.: A measure of association base on Gini's Mean difference. Communications in statistics. Theory and methods 16, 207–231 (1987)
22. Schindler, W.: A Combined Timing and Power Attack. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 263–279. Springer, Heidelberg (2002)
23. Schmidt, J.M., Tunstall, M., Avanzi, R.M., Kizhvatov, I., Kasper, T., Oswald, D.: Combined Implementation Attack Resistant Exponentiation. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 305–322. Springer, Heidelberg (2010)
24. Standaert, F.X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008)
25. Standaert, F.X., Gierlichs, B., Verbauwhede, I.: Partition *vs.* Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)
26. TELECOM ParisTech SEN research group: DPA Contest 2nd edn. (2009-2010), http://www.DPAcontest.org/v2/
27. Tufféry, S., Saporta, G.: Data mining et statistique décisionnelle. L'intelligence des données. Technip (2010); ISBN: 978271080946-3
28. Yitzhaki, S.: Gini's mean difference: a superior measure of variability for non-normal distributions. International Journal of Statistics 2, 285–316 (2003)

Two-Dimensional Representation of Cover Free Families and Its Applications: Short Signatures and More

Shota Yamada[1,⋆], Goichiro Hanaoka[2], and Noboru Kunihiro[1]

[1] The University of Tokyo
{yamada@it.,kunihiro@} k.u-tokyo.ac.jp
[2] National Institute of Advanced Industrial Science and Technology (AIST)
hanaoka-goichiro@aist.go.jp

Abstract. Very recently, Hofheinz, Jager, and Kiltz proposed novel digital signature schemes that yield significantly shorter signatures. However, in contrast to such remarkably short signatures, the size of the public key is still huge, making it desirable for this to be reduced. In this paper, we present a two-dimensional representation technique for cover free families, and show that this technique is quite useful for reducing the public key size in various cryptographic primitives. As immediate applications, we give constructions of the k-resilient identity-based key encapsulation mechanism (KEM), q-bounded CCA-secure KEM, and m-time signature which yield shorter public keys than previous schemes. Moreover, by applying our technique, we propose a (fully-fledged) signature scheme with the public key approximately 1/100 the size of that in the Hofheinz-Jager-Kiltz scheme with the same signature size and security assumption.

1 Introduction

Background. Designing more efficient basic cryptographic primitives, e.g., digital signatures, is one of the central research topics in cryptography. In performance evaluations of such primitives, the size of the public key is often considered to be less important since it does not need to be hidden from others and can even be kept in insecure storage. Nevertheless, it needs to be taken into account when it becomes extremely large. For example, if the size of a public key is significantly larger than the capacity of main memory in the device, the time to load the public key dominates the whole processing time.

Very recently, Hofheinz et al. [20] proposed novel digital signature schemes that yield significantly shorter signatures than the previously best known schemes (under reasonable assumptions). For a typical parameter setting, the size of their signature is only 200 bits long (for 80-bit security). However, in contrast to such surprisingly short signature length, the public key size is very large, approximately 26,000,000 bits for the same parameter settings. Therefore, it is

⋆ The first author is supported by a JSPS Research Fellowship for Young Scientists.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 260–277, 2012.

worth discussing the possibility of reducing the public key size in the Hofheinz-Jager-Kiltz scheme without increasing the signature size or strengthening the underlying security assumption.

Besides the Hofheinz-Jager-Kiltz scheme, there are many other cryptographic schemes that are very efficient except with respect to public key size (and thus, the computational cost of key generation). For example, Cramer et al. [9] proposed a q-bounded chosen-ciphertext secure public key encryption scheme whose ciphertext length is the same as that in the ElGamal scheme under the same security assumption, i.e., the decisional Diffie-Hellman (DDH) assumption. Thus, it is desirable to develop a general method for reducing the size of the public key in a certain class of cryptographic primitives (which includes the Hofheinz-Jager-Kiltz scheme).

Our Contribution. In this paper, we discuss a two-dimensional representation of a *cover free family*, and show that it is useful for reducing the size of the public key in a wide range of cryptographic schemes. An m-cover free family is a family of sets such that any m sets do not cover any another set. Each set in the family is a subset of $[d](= \{1, 2, \ldots, d\})$. Due to its combinatorial property, a cover free family has been used as a building block for constructions of many cryptographic protocols, such as [13,26,9,23,20], to name but a few. Some of the previous constructions, such as [9,20], associate one index $i \in [d]$ with one group element g^{a_i}, and consequently, these require at least d group elements in a public key. This is the main reason that the public key in cryptographic schemes reliant on a cover free family is generally huge.

In this work, to avoid the above problem, we introduce an m-cover free family over $[\sqrt{d}] \times [\sqrt{d}]$, which can easily be obtained from an ordinary cover free family over $[d]$. We call this representation of a cover free family, the *two-dimensional representation*. Then, we associate each *index of row* $i \in [\sqrt{d}]$ with one group element g^{a_i}, and each *index of column* $j \in [\sqrt{d}]$ with one group element g^{b_j}. Similarly, we associate index (i, j) (or a coefficient of the matrix) with $g^{a_i b_j}$.

With this two-dimensional representation of a cover free family (and a bilinear map), it is possible to compress the size of the public key in various cryptographic schemes that depend on cover free families. Specifically, in our schemes, we require only $2\sqrt{d}$ group elements in a public key, as opposed to d group elements in the previous scheme. A similar technique for reducing the public key size has also been used in [7,18,30], and our proposed technique can be considered to be an extension of this technique in cases where a cover free family is used.

As an immediate application of the above technique, a novel q-resilient identity based key encapsulation mechanism (IBKEM) with a very short ciphertext can be obtained. Specifically, its ciphertext length is only one group element, whereas previous (q-resilient and standard) IBKEM schemes [19,2,28,29] require at least two group elements in a ciphertext. Via the Canetti-Halevi-Katz (CHK) transformation [8,5,6] and the Naor transformation [4,11], we can obtain a novel key encapsulation mechanism (KEM) that is indistinguishable under q-bounded chosen ciphertext security (IND-q-CCA) [9] and a novel m-time signature scheme, respectively. In the above KEM, a ciphertext and a public key consist of only a

single group element and $O(q\sqrt{\lambda})$ group elements, respectively, and our scheme can be proven secure under the decisional bilinear Diffie-Hellman (DBDH) assumption. On the other hand, in the above signature scheme, a signature consists of only one group element with the size of its public key $O(m\sqrt{\lambda})$.

Finally, based on the two-dimensional representation of a cover free family, we demonstrate the construction of a fully-fledged signature scheme with very small signature size. More specifically, its signature is the same size as that of the Hofheinz-Jager-Kiltz scheme, and its public key size is approximately 1/100 of that of the Hofheinz-Jager-Kiltz scheme. For a typical parameter setting (with 80-bit security), our scheme yields signatures with only 200 bits and public keys with approximately 200,000 bits, while in the Hofheinz-Jager-Kiltz scheme, the signature size is the same, and the public key is approximately 26,000,000 bits long.

Related Works. Construction of a digital signature scheme with existential unforgeability under chosen message attack (EUF-CMA) [17] in the standard model is a main research topic in cryptography. In particular, the construction of a short signature from a mild assumption has been extensively studied. Earlier research proposed signature schemes from the strong RSA [10,16,15] and the strong q-Diffie Hellman (strong q-DH) assumptions [3]. Waters [28] proposed a signature scheme from the computational Diffie-Hellman (CDH) assumption. Many of these constructions implicitly or explicitly use a chameleon hash [24] to obtain a fully secure signature scheme. This conversion adds extra redundancy to the signature size. Hofheinz and Kiltz [21] introduced the notion of programmable hash functions and proposed generic constructions of short signatures using $(m, 1)$-programmable hash functions (see Section 2.3 for the definition). They also showed that Waters' hash [28] is indeed a $(2, 1)$-programmable hash function. As a result, from the strong q-DH and strong RSA assumptions, they obtained short signature schemes that do not require a chameleon hash. However, the construction of an $(m, 1)$-programmable hash function for $m \geq 3$, which could yield a shorter signature, was left as an open problem. Another recent progress was due to Hohenberger and Waters [22], who succeeded in constructing a signature scheme from the standard RSA assumption using a novel "prefix guessing" technique. Similar to most of the previous schemes, their construction uses a chameleon hash.

Very recently, Hofheinz, Jager, and Kiltz [20] solved the above open problem by constructing an $(m, 1)$-programmable hash function from an m-cover free family. They also removed the necessity of the chameleon hash from the RSA based signature in [22] using the technique from [21]. As a result, they obtained short signature schemes from the RSA assumption. Furthermore, they used the power of the $(1, \mathsf{poly})$-programmable hash function to obtain short signature schemes based on the (not strong, more standard) q-DH assumption. The scheme provides the shortest signatures in the literature, but with the drawback of huge public keys.

2 Preliminaries

For $\lambda \in \mathbb{N}$, 1^λ denotes the string of λ ones, with λ expressing the security parameter throughout this paper. $[\ell]$ denotes the set $\{1, 2, \ldots, \ell\}$. Moreover, $|x|$ and $|S|$ denote, respectively, the length of bitstring x, and the size of set S. If S is a set, $s \xleftarrow{\$} S$ denotes the action of uniform randomly selecting an element of S. Given algorithm $\mathcal{A}$, we write $z \xleftarrow{\$} \mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is a (probabilistic) algorithm that outputs z on input $(x, y, \ldots)$. We denote all generators in a group $\mathbb{G}$ by $\mathbb{G}^*$.

2.1 Number Theoretic Assumptions

Here we recall some number theoretic assumptions used in this paper.

Discrete Logarithm (DL) Assumption. Let $\mathbb{G}$ be a group of prime order p. We say that an adversary $\mathcal{A}$ (t, ϵ)-breaks the DL assumption on $\mathbb{G}$ if $\mathcal{A}$ runs in time t and $\Pr[\mathcal{A}(g, g^x) \xrightarrow{\$} x] \geq \epsilon$ where $g \xleftarrow{\$} \mathbb{G}$ and $x \xleftarrow{\$} \mathbb{Z}_p$. We assume that no algorithm exists that (t, ϵ)-breaks the DL assumption with polynomial t and non-negligible ϵ.

DBDH Assumption [6]. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order p with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We say that an adversary $\mathcal{A}$ (t, ϵ)-breaks the DBDH assumption on $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ if $\mathcal{A}$ runs in time t and $\frac{1}{2}|\Pr[\mathcal{A}(g_1, g_1^x, g_1^y, g_2, g_2^y, g_2^z, e(g, g)^{xyz}) \xrightarrow{\$} 0] - \Pr[\mathcal{A}(g_1, g_1^x, g_1^y, g_2, g_2^y, g_2^z, T) \xrightarrow{\$} 0]| \geq \epsilon$ where $g_1 \xleftarrow{\$} \mathbb{G}_1^*$, $g_2 \xleftarrow{\$} \mathbb{G}_2^*$, $T \xleftarrow{\$} \mathbb{G}_T$, $x, y, z \xleftarrow{\$} \mathbb{Z}_p$. We assume that no algorithm exists that (t, ϵ)-breaks the DBDH assumption with polynomial t and non-negligible ϵ.

q-DH Assumption. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order p with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We say that an adversary $\mathcal{A}$ (t, ϵ)-breaks the q-DH assumption on $\mathbb{G}_1$ and $\mathbb{G}_2$ if $\mathcal{A}$ runs in time t and $\Pr[\mathcal{A}(g_1, g_1^y, \ldots, g_1^{y^q}, g_2, g_2^y) \xrightarrow{\$} g_1^{1/y}] \geq \epsilon$ where $g_1 \xleftarrow{\$} \mathbb{G}_1^*$, $g_2 \xleftarrow{\$} \mathbb{G}_2^*$, $y \xleftarrow{\$} \mathbb{Z}_p^*$. We assume that no algorithm exists that (t, ϵ)-breaks the q-DH assumption with polynomial t and non-negligible ϵ.

2.2 Syntax and Security Notions

In this paper, we concentrate on the construction of an (IB)KEM scheme. Due to the KEM-DEM theorem [27], a q-resilient IBKEM and an IND-q-CCA secure KEM can be used as a q-resilient identity based encryption (IBE) and IND-q-CCA secure PKE, respectively by combining them with an appropriate DEM (data encapsulation mechanism).

IBKEM and Its q-Resilient Security. Here, we define the syntax of an IBKEM and its q-resilient security [19]. We assume that the upper bound of the number of users q is known a priori. Thus, Setup takes as input not only security parameter 1^λ, but also q. An IBKEM comprises four algorithms,

namely, $\mathsf{Setup}, \mathsf{KeyExtract}, \mathsf{Encapsulate}$, and $\mathsf{Decapsulate}$. We represent the ID space as $\mathcal{ID}$. The setup algorithm Setup generates the key pair $(PK, MSK) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, q)$ for master secret key MSK and public key PK. The key extraction algorithm $\mathsf{KeyExtract}$ inputs the public key, master secret key, and user ID and outputs a private key $SK_{ID} \xleftarrow{\$} \mathsf{KeyExtract}(PK, MSK, ID)$ for ID. The encapsulation algorithm $\mathsf{Encapsulate}$ takes a public key and ID as input and outputs the ciphertext for ID and its corresponding session key K as $(\psi, K) \xleftarrow{\$} \mathsf{Encapsulate}(PK, ID)$. The decapsulation algorithm $\mathsf{Decapsulate}$ takes a ciphertext and private key for ID as input and outputs K or $\perp = \mathsf{Decapsulate}(\psi, SK_{ID})$. K is the corresponding secret key and $\perp$ indicates that the ciphertext is not in a valid form. We require the usual correctness property.

We recall the q-resilient security experiment between a challenger and an adversary $\mathcal{A}$. First, the challenger runs $(PK, MSK) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, q)$ and the adversary is given PK. Proceeding adaptively, $\mathcal{A}$ requests private keys for $ID_1, \ldots, ID_q \in \mathcal{ID}$ under PK. The challenger responds to each query with a private key $SK_{ID_i} \xleftarrow{\$} \mathsf{KeyExtract}(PK, MSK, ID_i)$. $\mathcal{A}$ can interleave the key extraction queries with the challenge query at an arbitrary point. $\mathcal{A}$ queries for ID^* on which it wishes to be challenged. Then the challenger selects a random bit $b \in \{0, 1\}$ and runs $(\psi^*, K^*) \xleftarrow{\$} \mathsf{Encapsulate}(PK, ID^*)$. It sets $K_0 = K^*$ and chooses a random key $K_1 \xleftarrow{\$} \mathcal{K}$ from the key space $\mathcal{K}$. Then it gives (ψ^*, K_b) to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b and wins if $b = b'$.

In the above game, we constrain $\mathcal{A}$ so that $ID^* \notin \{ID_1, \ldots, ID_q\}$ to exclude a trivial attack. We define the advantage of $\mathcal{A}$ as $|\Pr[b = b'] - \frac{1}{2}|$. We say that $\mathcal{A}$ (t, ϵ)-breaks q-resilient security of the IBKEM if $\mathcal{A}$ runs in time t, makes at most q key extraction queries, and has advantage ϵ. We say that the IBKEM scheme is q-resilient secure if ϵ is negligible for any probabilistic polynomial-time algorithm $\mathcal{A}$, the number of key extraction queries of which is bounded by q.

KEM and its IND-q-CCA Security. The syntax of a KEM and its IND-q-CCA security is defined as in [9]. We assume that the upper bound of the number of decryption queries q is known a priori. Thus, a key generation algorithm takes q as input besides 1^λ. In the IND-q-CCA experiment, we constrain adversary $\mathcal{A}$ so that it does not issue the decryption query more than q times.

Digital Signature and Its EUF-CMA Security. We define the syntax of a digital signature scheme and its EUF-CMA security. A digital signature scheme is defined by the three algorithms, $\mathsf{Gen}, \mathsf{Sign}, and \mathsf{Verify}$. The key generation algorithm Gen generates a keypair $(PK, sk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ for a secret key sk and a public key PK. The signing algorithm Sign inputs a message and the secret key, and returns a signature $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, M)$ of the message. The verification algorithm Verify takes a public key and a message with a corresponding signature as input, and returns $\top$ or $\perp$, indicating "accept" or "reject", respectively. We require the usual correctness properties.

We recall the EUF-CMA experiment played by a challenger and a forger $\mathcal{F}$. First, the challenger runs $(PK, sk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and $\mathcal{F}$ is given PK. Proceeding adaptively, $\mathcal{F}$ requests signatures on messages $M_1, \ldots, M_q \in \{0, 1\}^*$ under PK.

The challenger responds to each query with a signature $\sigma_i \xleftarrow{\$} \mathsf{Sign}(sk, M_i)$. Eventually, $\mathcal{F}$ outputs the pair (M^*, σ^*). We say that the adversary wins the game if $\mathsf{Verify}(M^*, \sigma^*, PK) = \top$ and $M^* \notin \{M_1, \ldots, M_q\}$. We say that $\mathcal{F}$ (t, q, ϵ)-breaks the EUF-CMA security of the signature if $\mathcal{F}$ runs in time t, makes at most q signing queries, and has success probability ϵ. We say that the signature scheme is EUF-CMA secure if ϵ is negligible for any probabilistic polynomial-time algorithm $\mathcal{F}$. If we include m as input for Gen and limit the adversary $\mathcal{A}$ in the above game so that the number of signing queries is less than m, then the above definition corresponds to the definition of an m-time signature.

2.3 Programmable Hash Functions ([21])

Let $\mathbb{G}$ be a group of known order p. A group hash scheme D over $\mathbb{G}$ with input length l is associated with two efficient algorithms $\mathsf{PHF.Gen}$ and $\mathsf{PHF.Eval}$. The probabilistic algorithm $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}$ generates a hash key κ for security parameter λ. $\mathsf{PHF.Eval}$ is a deterministic algorithm that takes as input a hash function key κ and $s \in \{0,1\}^l$ and returns $\mathsf{PHF.Eval}(\kappa, s) \in \mathbb{G}$. In fact, our scheme does not use $\mathsf{PHF.Gen}$, but we include it here for completeness of the programmable hash function definition.

Definition 1. *A group hash scheme* $\mathsf{D} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ *is* (m, n, γ, δ)*-programmable, if there exists an efficient trapdoor key generation algorithm* $\mathsf{PHF.TrapGen}$ *and an efficient trapdoor evaluation algorithm* $\mathsf{PHF.TrapEval}$ *with the following properties.*

- *The probabilistic algorithm* $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ *generates hash function key* κ *together with trapdoor information* τ *given security parameter* λ *and* $g, h \in \mathbb{G}$.
- *For all* $g, h \in \mathbb{G}^*$, *the keys* $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda)$ *and* $\kappa' \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ *are statistically* γ*-close.*
- *On input* $s \in \{0,1\}^l$ *and trapdoor information* τ, *the deterministic trapdoor evaluation algorithm* $(e_s, f_s) \leftarrow \mathsf{PHF.TrapEval}(\tau, s)$ *produces* $e_s, f_s \in \mathbb{Z}_p$ *so that for all* $s \in \{0,1\}^l$, $\mathsf{PHF.Eval}(\kappa, s) = g^{e_s} h^{f_s}$. *We denote* $\mathsf{D}(s) = \mathsf{PHF.Eval}(\kappa, s)$.
- *For all* $g, h \in \mathbb{G}$, *all* κ *output by* $\kappa \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda, g, h)$ *and all* $s_1^*, \ldots, s_m^* \in \{0,1\}^l$ *and* $s_1, \ldots, s_n \in \{0,1\}^l$ *such that* $s_i^* \neq s_j$ *for all* i, j, *we have* $\Pr[e_{s_1^*} = \cdots = e_{s_m^*} = 0 \wedge e_{s_1}, \ldots, e_{s_n} \neq 0] \geq \delta$, *where* $(e_{s_i^*}, f_{s_i^*}) = \mathsf{PHF.TrapEval}(\tau, s_i^*)$ *and* $(e_{s_j}, f_{s_j}) = \mathsf{PHF.TrapEval}(\tau, s_j)$ *and the probability is taken over the trapdoor* τ *produced together with* κ.

If γ *is negligible and* δ *is noticeable, we say that* D *is* (m, n)*-programmable for short. Moreover, if* D *is* $(1, q)$*-programmable for every polynomial* $q = q(\lambda)$, *we say that* D *is* $(1, \mathsf{poly})$*-programmable.*

We note that Waters' hash [28] is a concrete example of a $(1, \mathsf{poly})$-programmable hash function.

3 Our Basic Idea and Its Direct Applications

In this section, we first recall the standard definition of a cover free family. Then, we introduce a slight twist to the use of cover free families and propose

a novel q-resilient IBKEM under the DBDH assumption as a direct application of our technique. The ciphertext and user private key of q-resilient IBKEM consist of only one group element. Next, we apply the CHK transformation [8,5,6] and Naor transformation [4,11] for the scheme to obtain a new IND-q-CCA secure KEM and a multiple-time signature, respectively. A public key of the resulting KEM consists of $O(q\sqrt{\lambda})$ group elements, which is shorter than that of the previous IND-q-CCA secure KEM [9], which consists of $O(q^2\lambda)$ group elements. The public key of our multiple-time signature consists of $O(m\sqrt{\lambda})$ group elements, which is shorter than that in most of the previous multiple-time signature schemes based on a cover free family [26,20,12], consisting of $O(m^2\lambda)$ group elements.

3.1 Two-Dimensional Representation of Cover Free Family

We begin by recalling the definition of cover-free families. Let S_1, S_2 be sets. We say that S_2 does not cover S_1 if $S_1 \not\subseteq S_2$. Let d, m, α be integers, and let $F = (F_\mu)_{\mu\in[\alpha]}$ be a family of α subsets of $[d]$. We say that F is m-cover free if for any set I containing (up to) m indices $I = \{\mu_1, \ldots, \mu_m\} \subseteq [\alpha]$, it holds that $F_\nu \not\subseteq \cup_{\mu\in I} F_\mu$ for any ν that is not contained in I. In other words, if $|I| \leq m$, then the union $\cup_{\mu\in I} F_\mu$ does not cover F_ν for all $\nu \in [\alpha]\backslash I$. We say that F is w-uniform if $|F_\mu| = w$ for all $\mu \in [\alpha]$. Throughout this paper, we use a parameter in the following lemma.

Lemma 1. *([14,25]) There is a deterministic polynomial-time algorithm that, on input of integers $m, \alpha = 2^n$, returns $d \in \mathbb{N}$ and the set family $F = (F_\mu)_{\mu\in[\alpha]}$, such that F is m-cover free over $[d]$ and w-uniform, where $d \leq 16m^2n$ and $w = d/4m$.*

Here, we introduce our novel technique for the use of a cover free family. In this paper, we regard $[d]$ as $[\ell_1] \times [\ell_2]$, where ℓ_1 and ℓ_2 are integers satisfying $\ell_1 \geq \ell_2$ and $\ell_1\ell_2 \geq d$. We regard $i \in [d]$ as an element of $[\ell_1] \times [\ell_2]$ by associating it with $(i - \ell_1(\lceil i/\ell_1 \rceil - 1), \lceil i/\ell_1 \rceil)$. Then, all F_μ can be seen as a subset of $[\ell_1] \times [\ell_2]$ in a natural way and $(F_\mu)_{\mu\in\alpha}$ can be seen as an m-cover free family over $[\ell_1] \times [\ell_2]$.

In our construction, we associate an element $D \in \mathcal{D}$ of some domain $\mathcal{D}$ with a subset of $[\ell_1] \times [\ell_2]$ by defining $S : \mathcal{D} \to 2^{[\ell_1]\times[\ell_2]}$ as $S(D) \stackrel{def}{=} F_{H(D)} \subseteq [\ell_1] \times [\ell_2]$, where $F = (F_\mu)_{\mu\in[\alpha]}$ is an m-cover free family over $[\ell_1] \times [\ell_2]$ and $H : \mathcal{D} \to [\alpha]$ is an injective (or hash) function. In the following, we treat H as an injective function for simplicity, but it is enough to assume that H is a collision resistant hash for our schemes to be secure. To avoid a birthday attack, we typically set $n = 2\lambda$. From the property of F, for all $D^*, D_1, \ldots, D_m$ it holds that $S(D^*) \not\subseteq \cup_{i=1}^m S(D_i)$ if $D^* \notin \{D_1, \ldots, D_m\}$. Besides, if we require F to be w-uniform, then $|S(D)| = w$ for all $D \in \mathcal{D}$.

3.2 q-Resilient IBKEM

Here, we show the construction of a q-resilient IBKEM from the DBDH assumption using our technique explained above. Since a q-resilient IBKEM is a

powerful cryptographic protocol, it can be used as a building block for various cryptographic schemes such as the IND-q-CCA KEM and multiple-time signature scheme via the CHK transformation [8,5,6] and Naor transformation [4,11], respectively. Our IBKEM can be seen as a modification of the IBKEM implicit in [9] from the DDH assumption. The ciphertext of our IBKEM consists of only one group element, and it is smaller than any previous (q-resilient and standard) IBKEM schemes [19,2,28,29]. This is a remarkable property of our scheme.

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order p with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Typically, we set $(\ell_1, \ell_2) = (O(q\sqrt{\lambda}), O(q\sqrt{\lambda}))$. Let S be a map $S : \mathcal{ID} \to 2^{[\ell_1]\times[\ell_2]}$ where $\mathcal{ID}$ is the ID space of the scheme. We assume that for all $ID^*, ID_1, \ldots, ID_q \in \mathcal{ID}$ it holds that $S(ID^*) \not\subseteq \cup_{i=1}^{q} S(ID_i)$ if $ID^* \notin \{ID_1, \ldots, ID_q\}$. We define the scheme as follows.

$\mathsf{Setup}(1^\lambda, q)$**:** This selects $g_1 \xleftarrow{\$} \mathbb{G}_1^*$, $g_2 \xleftarrow{\$} \mathbb{G}_2^*$, $a_1, \ldots, a_{\ell_1}, b_1, \ldots, b_{\ell_2} \xleftarrow{\$} \mathbb{Z}_p$ and computes $A_1 = g_1^{a_1}, \ldots, A_{\ell_1} = g_1^{a_{\ell_1}}, B_1 = g_2^{b_1}, \ldots, B_{\ell_2} = g_2^{b_{\ell_2}}$. Then, it returns the public key $PK = (g_1, g_2, A_1, \ldots, A_{\ell_1}, B_1, \ldots, B_{\ell_2})$ and master secret key $MSK = (a_1, \ldots, a_{\ell_1}, b_1, \ldots, b_{\ell_2})$.

$\mathsf{KeyExtract}(PK, MSK, ID)$**:** This computes $SK_{ID} = g_2^{\sum_{(i,j)\in S(ID)} a_i b_j}$ and returns it.

$\mathsf{Encapsulate}(PK, ID)$**:** This first selects $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $\psi = g_1^r$, $K = \left(\prod_{(i,j)\in S(ID)} e(A_i, B_j)\right)^r$. Then, it returns ciphertext ψ and its corresponding key K.

$\mathsf{Decapsulate}(\psi, PK, SK_{ID})$**:** This computes $K' = e(\psi, SK_{ID})$ and returns it.

Remark. In the Encapsulate algorithm, we need at most ℓ_2 (not $|S(ID)|$) pairing computations to compute K. To confirm this, it is enough to check that $K = \left(\prod_{j\in[\ell_2]} e\big(\prod_{i\in\{i|(i,j)\in S(ID)\}} A_i, B_j\big)\right)^r$. A similar technique can be applied to other constructions in this paper. We also note that the above and all other schemes in this paper can also be implemented in symmetric pairing groups where $\mathbb{G}_1 = \mathbb{G}_2$. The resulting schemes are secure under the corresponding number theoretic assumptions in the symmetric pairing group.

The following theorem establishes the security of the scheme.

Theorem 1. *Suppose there exists an adversary $\mathcal{A}$ that (t, q, ϵ)-breaks the q-resilient security of the above IBKEM scheme. Then there exists an adversary $\mathcal{B}$ that (t', ϵ')-breaks the DBDH assumption on $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ with $t' \approx t$ and $\ell_1\ell_2\epsilon' \geq \epsilon$.*

Proof. Let ID_k be the k-th query to the key extraction oracle and ID^* be the challenge ID. In the following, let X_i denote that $\mathcal{A}$'s guess is correct (i.e., $b' = b$) in Game i. We consider the following games.

Game **0.** We define Game 0 as the q-resilient security experiment between a challenger and an adversary $\mathcal{A}$. By definition we have $|\Pr[X_0] - 1/2| = \epsilon$.

Game **1.** Game 1 proceeds like Game 0. The only difference is that at the beginning of the game, the challenger chooses $(i^*, j^*) \xleftarrow{\$} [\ell_1] \times [\ell_2]$. Since nothing has essentially been changed, we have $\Pr[X_1] = \Pr[X_0]$.

Game 2. Game 2 proceeds like Game 1, but we introduce some notations for later use. Let $Q \subseteq [\ell_1] \times [\ell_2]$ be $Q = \cup_{k=1}^{q} S(ID_k)$. From the property of S, we have $S(ID^*) \not\subseteq Q$. Thus we can define $(i^*_{min}, j^*_{min}) \stackrel{def}{=} \min(S(ID^*) \backslash Q)$ where min is the minimum function based on some well-defined order (e.g., lexicographic order) over $[\ell_1] \times [\ell_2]$. Call Fail the event that $(i^*, j^*) \neq (i^*_{min}, j^*_{min})$. Note that $\Pr[\mathsf{Fail}|X_2] = (\ell_1\ell_2 - 1)/\ell_1\ell_2 = \Pr[\mathsf{Fail}]$ so X_2 and Fail are independent events, and in particular, $\Pr[X_2] = \Pr[X_2|\neg\mathsf{Fail}]$. Since we did not actually change anything, $\Pr[X_2] = \Pr[X_1]$.

Game 3. In Game 3, we substitute $\mathcal{A}$'s output b' with a random bit whenever Fail occurs. Obviously, $\Pr[X_3|\neg\mathsf{Fail}] = \Pr[X_2|\neg\mathsf{Fail}]$ and $\Pr[X_3|\mathsf{Fail}] = \frac{1}{2}$. Since $\Pr[\mathsf{Fail}] = (\ell_1\ell_2 - 1)/\ell_1\ell_2$ in Game 3 as well, we can establish that $|\Pr[X_3] - 1/2| = |\Pr[X_2] - 1/2|/\ell_1\ell_2$.

Game 4. In Game 4, we immediately stop the experiment and set Fail to true (hence immediately taking a random bit for $\mathcal{A}$'s output) as soon as $\mathcal{A}$ asks for a key for ID_k such that $(i^*, j^*) \in S(ID_k)$ or $(i^*, j^*) \notin S(ID^*)$. Note that even in Game 3, such a query would imply $(i^*_{min}, j^*_{min}) \neq (i^*, j^*)$ and hence Fail. Consequently, $\Pr[X_4] = \Pr[X_3]$.

Proving the following lemma completes the proof. □

Lemma 2. *There exists an adversary $\mathcal{B}$ that (t', ϵ')-breaks the DBDH assumption with $t' \approx t$ and $\epsilon' \geq |\Pr[X_4] - 1/2|$.*

Proof. **DBDH Adversary.** We replace the challenger with DBDH adversary $\mathcal{B}$ with advantage $|\Pr[X_4] - \frac{1}{2}|$. $\mathcal{B}$ receives a DBDH challenge $(g_1, g_1^x, g_1^y, g_2, g_2^y, g_2^z, R)$ as input and tries to distinguish whether $R = e(g_1, g_2)^{xyz}$ or R is a random element of $\mathbb{G}_T$.

Setup of Public Keys. $\mathcal{B}$ first sets $g_1 = g_1, g_2 = g_2$ and chooses $i^* \stackrel{\$}{\leftarrow} [\ell_1]$, $j^* \stackrel{\$}{\leftarrow} [\ell_2]$. Then, it chooses $a_i \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ for all $i \neq i^*$ and $b_j \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ for all $j \neq j^*$. The public key is set as $A_i = g_1^{a_i}$ for all $i \neq i^*$, $A_{i^*} = g_1^y$, $B_j = g_2^{b_j}$ for all $j \neq j^*$, and $B_{j^*} = g_2^z$. Note that $\mathcal{B}$ implicitly sets $a_{i^*} = y, b_{j^*} = z$. Then, $\mathcal{B}$ gives $PK = (g_1, g_2, A_1, \ldots, A_{\ell_1}, B_1, \ldots, B_{\ell_2})$ to $\mathcal{A}$.

Setup of Challenge Ciphertext. At some point, $\mathcal{B}$ outputs ID^*. If $(i^*, j^*) \notin S(ID^*)$, $\mathcal{A}$ aborts and outputs a random bit. Otherwise, $\mathcal{B}$ sets $\psi = g_1^x$ and computes K by $K = R \cdot \prod_{(i,j) \in S(ID^*) \backslash \{(i^*, j^*)\}} K_{i,j}$. Note that $\mathcal{B}$ sets $r = x$ implicitly. Here, the $K_{i,j}$ are computed as follows:

$$K_{i,j} = \begin{cases} e(g_1^x, g_2)^{a_i b_j} = \big(e(A_i, B_j)\big)^r & (i \neq i^*, j \neq j^*) \\ e(g_1^x, g_2^y)^{b_j} = \big(e(A_{i^*}, B_j)\big)^r & (i = i^*, j \neq j^*) \\ (g_1^x, g_2^z)^{a_i} = \big(e(A_i, B_{j^*})\big)^r & (i \neq i^*, j = j^*). \end{cases}$$

Then, $\mathcal{B}$ gives (ψ, K) to $\mathcal{A}$. If $R = e(g_1, g_2)^{xyz}$, $R = e(A_{i^*}, B_{j^*})^r$ and K is the corresponding key for the challenge ciphertext ψ. If R is a random element of $\mathbb{G}_T$, K is distributed uniform randomly in $\mathbb{G}_T$.

Answering Key Extraction Queries. When $\mathcal{A}$ makes a key extraction query for ID_k, $\mathcal{B}$ first checks whether $(i^*, j^*) \in S(ID_k)$. If so, $\mathcal{B}$ aborts and outputs a random bit. Otherwise, $\mathcal{B}$ computes SK_{ID} by $SK_{ID} = \prod_{(i,j)\in S(ID_k)} sk_{i,j}$ where the $sk_{i,j}$ are computed as follows:

$$sk_{i,j} = \begin{cases} g_2^{a_i b_j} & (i \neq i^*, j \neq j^*) \\ (g_2^y)^{b_j} = g_2^{a_{i^*} b_j} & (i = i^*, j \neq j^*) \\ (g_2^z)^{a_i} = g_2^{a_i b_{j^*}} & (i \neq i^*, j = j^*). \end{cases}$$

Guessing the Answer to DBDH Challenge. $\mathcal{A}$ outputs its guess b'. Then, $\mathcal{B}$ checks whether $(i^*_{min}, j^*_{min}) = (i^*, j^*)$. If so, $\mathcal{B}$ outputs the same b' as its guess. Otherwise, $\mathcal{B}$ outputs a random bit.

By the construction, it is easy to see that $|\Pr[\mathcal{B}'\text{s guess is correct}] - 1/2| = |\Pr[X_4] - 1/2|$. □

3.3 IND-q-CCA Secure KEM with Shorter Public Keys

By applying the CHK transformation [8,5,6] to our IBKEM, we can obtain a novel IND-q-CCA secure KEM. The security of our scheme can be proven under the DBDH assumption. In our scheme, a ciphertext consists of only one group element, while a public key consists of only $O(q\sqrt{\lambda})$ group elements, which is less than the $O(q^2\lambda)$ group elements required by the previous IND-q-CCA scheme [9].

Here, we compare our scheme with the IND-q-CCA secure KEM scheme in [9] under the DDH assumption. A ciphertext in the latter scheme consists of only one group element, which is the same as our construction. As for public and secret key size, however, the latter construction is $O(q^2\lambda)$, whereas our construction is $O(q\sqrt{\lambda})$. This difference has an important implication, in that, if the q operation is infeasible in practice, then it might be unnatural to assume that a user can generate the $O(q^2\lambda)$ key pairs required in [9], as opposed to only $O(q\lambda)$ key pairs required in our construction. Reducing the computational cost of setup to be $O(q)$ at the cost of security under a stronger number theoretic assumption (i.e., DBDH) than that of the previous construction (i.e., DDH) can be seen as a partial solution to this problem.

3.4 Multiple-Time Signature with Shorter Public Key

Similarly, by applying the Naor transformation [4,11] to our IBKEM, we can obtain a novel multiple-time signature. The security of our scheme can be proven under the CDH assumption. In the resulting scheme, a signature consists of only one group element (in $\mathbb{G}_1$) while the public key consists of $O(m\sqrt{\lambda})$ group elements if we set the parameters appropriately. No other scheme achieves simultaneously the same signature size and public key size as our scheme.

The signature size of our scheme is $O(1)$, while the public and secret key size is $O(m)$. (Here, we ignore the dependency on λ and focus on the dependency

on m.) We compare our scheme with previous multiple-time signature schemes [26,31,12,20] based on the cover free family technique. In most of these schemes, the public key consists of $O(m^2)$ elements. The only scheme that achieves an $O(1)$ signature size and $O(m)$ public key size simultaneously is given in [20]. Such a scheme can be obtained by combining a $(m, 1)$ weak programmable hash function in [20] with a chameleon hash as suggested in [20]. If we choose prime order group $\mathbb{G}$ as the underlying group, the scheme can be proven secure under the DL assumption. Although the other scheme can be proven secure under a weaker assumption than ours, the signature length of that scheme is longer than ours due to the use of a chameleon hash. We also note that there is no known (fully fledged, not m-time) signature scheme that achieves the same signature size as ours in the standard model.

4 Short Signature with Smaller Public Key Size

So far, we have seen several direct applications of our technique. In this section, we show a further complicated and non-trivial application of our technique. Our ultimate goal is the construction of a digital signature scheme with a short parameter under a mild assumption. Toward achieving this goal, we present a signature scheme with very short signatures and smaller public key size than the previous construction [20]. Our starting point is the signature scheme proposed very recently in [20] ($\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{cfs}}, \mathsf{H}_{\mathsf{wat}}]$). This construction provides the shortest signatures yet in the literature and can be proven secure under the q-DH assumption. While providing very short signatures, the scheme suffers from a large public key size. This inefficiency is mainly due to the use of an $(m, 1)$-programmable hash function. Thus, we remove the necessity of the $(m, 1)$-programmable hash function from the construction and show that the scheme remains secure if we use our m time signature scheme instead. By this modification, we can reduce the public key size of the scheme, while preserving its security and signature size.

4.1 Construction

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order p with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Typically, we set $(\ell_1, \ell_2) = (O(m\sqrt{\lambda}), O(m\sqrt{\lambda}))$. Let $D : \{0,1\}^l \to \mathbb{G}_2$ be a $(1, \mathsf{poly})$-programmable hash function with algorithms $(\mathsf{PHF.Gen}, \mathsf{PHF.Eval}, \mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval})$. Let S be a map $S : \mathcal{M} \to 2^{[\ell_1]\times[\ell_2]}$ where $\mathcal{M}$ is the message space. We assume that for all $M^*, M_1, \ldots, M_m \in \mathcal{M}$ it holds that $S(M^*) \not\subseteq \cup_{i=1}^{m} S(M_i)$ if $M^* \notin \{M_1, \ldots, M_m\}$. We define the scheme as follows.

$\mathsf{Gen}(1^\lambda)$**:** This first selects $g_1 \xleftarrow{\$} \mathbb{G}_1^*$, $g_2 \xleftarrow{\$} \mathbb{G}_2^*$, and then generates a hash function through $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g_2, g_2^y)$, where $y \xleftarrow{\$} \mathbb{Z}_p$. Then, it selects $a_1, \ldots, a_{\ell_1}, b_1, \ldots, b_{\ell_2} \xleftarrow{\$} \mathbb{Z}_p$ and computes $A_1 = g_1^{a_1}, \ldots, A_{\ell_1} = g_1^{a_{\ell_1}}$, $B_1 = g_2^{b_1}, \ldots, B_{\ell_2} = g_2^{b_{\ell_2}}$. It returns public key $PK = (\kappa, g_1, g_2, A_1, \ldots, A_{\ell_1}, B_1, \ldots, B_{\ell_2})$ and secret key $sk = (y, \tau, a_1, \ldots, a_{\ell_1}, b_1, \ldots, b_{\ell_2})$.

In the following, $\mathsf{D}(s)$ denotes $\mathsf{PHF.Eval}(\kappa, s)$ and $\mathsf{d}(s) \in \mathbb{Z}_p$ denotes $\log_{g_2} \mathsf{D}(s)$ (i.e., $g_2^{\mathsf{d}(s)} = \mathsf{D}(s)$).

$\mathsf{Sign}(sk, M)$**:** This computes $\mathsf{H}(M) = g_1^{\sum_{(i,j)\in S(M)} a_i b_j}$. Then it selects a random $s \xleftarrow{\$} \{0,1\}^l$ until $\mathsf{d}(s) \neq 0$ and computes $\sigma = \mathsf{H}(M)^{1/\mathsf{d}(s)}$. [1] Since $\mathsf{d}(s) = e_s + y f_s$ where $(e_s, f_s) = \mathsf{PHF.TrapEval}(\tau, s)$, $\mathsf{d}(s)$ can be efficiently computed from y and τ. The signature is $(\sigma, s) \in \mathbb{G}_1 \times \{0,1\}^l$.

$\mathsf{Verify}(M, (\sigma, s), PK)$**:** This returns $\top$ if $e(\sigma, \mathsf{D}(s)) = \prod_{(i,j)\in S(M)} e(A_i, B_j)$. Otherwise, it returns $\bot$.

Remark. Signature σ can be computed by $\sigma = g_1^{\sum_{(i,j)\in S(M)} a_i b_j / \mathsf{d}(s)}$ instead of first computing $\mathsf{H}(M)$ and then computing $\sigma = \mathsf{H}(M)^{1/\mathsf{d}(s)}$. The exponentiation needed to compute the signature occurs only once. Also note that $\mathsf{H}(M)$ is our multiple-time signature in Section 3.4. Since $\mathsf{H}(M)$ cannot be computed efficiently without a secret key, H is not an $(m, 1)$-programmable hash function. Thus our construction above does not fall into a special case of the generic construction in [20], which uses an $(m, 1)$-programmable hash function.

4.2 Security

Theorem 2. *Let* D *be* $(1, \mathsf{poly}, \gamma, \delta)$*-programmable. Suppose there exists a forger* $\mathcal{F}$ *that* (t, q, ϵ)*-breaks the EUF-CMA security of the above scheme. Then there exists an adversary* $\mathcal{A}$ *that* (t', ϵ')*-breaks the q-DH assumption on* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *with* $t \approx t'$ *and* $\epsilon \leq \frac{1+q\ell_1\ell_2}{\delta}\epsilon' + \frac{q^{m+1}}{2^{ml}}$.

In the rest of this subsection, we prove Theorem 2. Before going into the details, we give an intuitive explanation of the security proof of our scheme. The high level structure of the security proof is very similar to that in [20]. Let $M_k, (s_k, \sigma_k)$ be the k-th signing query of forger $\mathcal{F}$ and the response to it. Let $M^*, (s^*, \sigma^*)$ be $\mathcal{F}$'s final output.

We consider two types of forgers, Type1 and Type2, as in [20]. A type1 forger reuses s_k. That is, $s^* = s_k$ for some $k \in [q]$. A type2 forger outputs a forgery such that $s^* \neq s_k$ for all $k \in [q]$. We construct a q-DH adversary $\mathcal{A}$ from the forger against our scheme. To deal with Type1 forgers, since $|\{k | s_k = s^*\}| \geq 1$, $\mathcal{A}$ has to compute signatures of the form $\mathsf{H}(M_k)^{1/\mathsf{d}(s^*)}$ more than once. In fact, due to the generalized birthday bound by [20] (see Lemma 3 below), $\Pr[|\{k | s_k = s^*\}| \geq m + 1]$ is negligible. Thus, it is sufficient if $\mathcal{A}$ can compute a signature with form $\mathsf{H}(M_k)^{1/\mathsf{d}(s^*)}$, m times. To handle this problem, [20] used the power of the $(m, 1)$-programmable hash function. Conversely, we use the power of the m time signature scheme. The security of the scheme is essentially reduced to the security of the multiple-time signature. The reduction for a Type2 forger is more

[1] This step makes Sign run in the expected polynomial time. To avoid this, we can modify Sign so that when $\mathsf{d}(s) = 0$, it outputs secret key sk as suggested in [20]. This change does not harm the security of the scheme since the probability of $\mathsf{d}(s) = 0$ is negligible as proven in the security proof of the scheme.

similar to the security proof of previous schemes. Here, we recall the following lemma from [20], which is needed in our security proof.

Lemma 3. *Let A be a set with $|A| = a$. Let $X_1, \ldots, X_q$ be q independent random variables, with uniformly random values taken from A. Then, the upper bound on the probability that there exists $m+1$ pairwise distinct indices $i_1, \ldots, i_{m+1}$ such that $X_{i_1} = \cdots = X_{i_{m+1}}$ is $\frac{q^{m+1}}{a^m}$.*

Theorem 2 follows directly from Lemmas 4 and 5, proving security against Type2 and Type1 forgers, respectively. We omit the proof of Lemma 5 due to a lack of space.

Lemma 4. *Let $\mathcal{F}$ be a Type1 forger that (t, q, ϵ)-breaks the existential unforgeability of our scheme. Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the q-DH assumption on $\mathbb{G}_1$ and $\mathbb{G}_2$ with $t \approx t'$ and $\epsilon' \geq \frac{\delta}{1+q\ell_1\ell_2}(\epsilon - \frac{q^{m+1}}{2^{ml}})$.*

Lemma 5. *Let $\mathcal{F}$ be a Type2 forger that (t, q, ϵ)-breaks the existential unforgeability of the above scheme. Then there exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the q-DH assumption with $t \approx t'$ and $\epsilon' \geq \frac{\delta\epsilon}{3+\delta p/(p-1)}$.*

Proof. (of Lemma 4) In the following, let X_i denote the probability that $\mathcal{F}$ is successful in Game i and the challenger does not abort.

Game 0. We define Game 0 as the EUF-CMA experiment between a challenger and forger $\mathcal{F}$. By definition we have $\Pr[X_0] = \epsilon$.

Game 1. In this game, the challenger aborts if there exist at least $m+1$ indices $k_1, \ldots, k_{m+1} \in [q]$ such that $s_k = s_{k'}$ for all $k, k' \in \{k_1, \ldots, k_{m+1}\}$. We denote this event by $\mathsf{Abort}_{\mathsf{mColl}}$. We know that $\Pr[\mathsf{Abort}_{\mathsf{mColl}}] \leq \frac{q^{m+1}}{2^{ml}}$ from Lemma 3. Thus, we have $\Pr[X_1] \geq \Pr[X_0] - \frac{q^{m+1}}{2^{ml}}$.

Game 2. In this game, the challenger chooses randomness $s_1, \ldots, s_q$ in advance and aborts if $\mathsf{D}(s_k) = 1_{\mathbb{G}_2}$ (which means $\mathsf{d}(s_k) = 0$) for some $k \in [q]$. We denote this event as $\mathsf{Abort}_{\mathsf{Dzero}}$. Then we have $\Pr[X_2] \geq \Pr[X_1] - \Pr[\mathsf{Abort}_{\mathsf{Dzero}}]$.

Game 3. In this game, the challenger guesses $k^* \xleftarrow{\$} [q]$ such that $s_{k^*} = s^*$ in advance and aborts if $\mathcal{F}$ outputs a forgery (M^*, σ^*, s^*) with $s_{k^*} \neq s^*$. Since $s^* \in \{s_i\}_{i=1}^q$, we have $\Pr[X_3] \geq \Pr[X_2]/q$.

Game 4. In this game, the challenger chooses $i^* \xleftarrow{\$} [\ell_1]$, $j^* \xleftarrow{\$} [\ell_2]$ before setting the public key and aborts if $(i^*, j^*) \notin S(M^*)$ or $(i^*, j^*) \in S(M_k)$ for some $k \in \{k \mid s_k = s^*\}$. Recall that $\{k \mid s_k = s^*\} \leq m$, so $S(M^*) \not\subseteq \cup_{k \in \{k \mid s_k = s^*\}} S(M_k)$ from the property of S. Thus, there exists at least one $(i', j') \in [\ell_1] \times [\ell_2]$ such that $(i', j') \in S(M^*)$ and $(i', j') \notin S(M_k)$ for all $k \in \{k \mid s_k = s^*\}$. We have $\Pr[X_4] \geq \Pr[X_3]/\ell_1\ell_2$.

Game 5. In the following, let $E = \cup_{i=1}^q \{s_i\}$, $E^* = E \backslash \{s_{k^*}\}$. In this game the challenger sets $A_i = g_1^{\tilde{a}_i} \prod_{t \in E} \mathsf{d}(t)$ for all $i \neq i^*$, and $A_{i^*} = g_1^{\tilde{a}_{i^*}} \prod_{t \in E^*} \mathsf{d}(t)$. The challenger also sets $B_j = g_2^{\mathsf{d}(s_{k^*})\tilde{b}_j}$ for all $j \neq j^*$, and $B_{j^*} = g_2^{\tilde{b}_{j^*}}$. Here, $\tilde{a}_1, \ldots, \tilde{a}_{\ell_1}, \tilde{b}_1, \ldots, \tilde{b}_{\ell_2} \xleftarrow{\$} \mathbb{Z}_p$. Since $\mathsf{d}(t) \neq 0$ for all $t \in E$, the distribution of the public key is unchanged from the previous game. Since this change is only conceptual, we have $\Pr[X_5] = \Pr[X_4]$.

Game 6. In this game, the challenger computes $(e_{s_k}, f_{s_k}) \leftarrow \mathsf{PHF.TrapEval}(\tau, s_k)$ for $k \in [q]$. In the following, let $(e^*, f^*) = (e_{s^*}, f_{s^*}) \leftarrow \mathsf{PHF.TrapEval}(\tau, s^*)$. The challenger aborts if $e_{s_k} = 0$ for some $k \in [q]$ such that $s_k \neq s^*$ or $e^* \neq 0$. Note that if $\mathsf{d}(s^*) = e^* + yf^* \neq 0$ and $e^* = 0$, then $f^* \neq 0$. By $(1, \mathsf{poly})$-programmability of D, we have $\Pr[X_6] \geq \delta \Pr[X_5]$.

Proving Claims 1 and 2 below completes the proof. We omit the proof of Claim 1 since it can be proven easily. □

Claim 1. *There exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the DL assumption on $\mathbb{G}_2$ (and thus, (t', ϵ')-breaks the q-DH assumption on $\mathbb{G}_1$ and $\mathbb{G}_2$) with $t' \approx t$ and $\epsilon' \geq \delta \Pr[\mathsf{Abort}_{\mathsf{Dzero}}]$.*

Claim 2. *There exists an adversary $\mathcal{A}$ that (t', ϵ')-breaks the q-DH assumption on $\mathbb{G}_1$ and $\mathbb{G}_2$ with $t' \approx t$ and $\epsilon' \geq \Pr[X_6]$.*

Proof. (of Claim 2) **q-DH adversary.** We replace the challenger in Game 6 with q-DH adversary $\mathcal{A}$ whose advantage is $\Pr[X_6]$. $\mathcal{A}$ receives a q-DH challenge $(g_1, g_1^y, \ldots, g_1^{y^q}, g_2, g_2^y)$ as input and tries to compute $g_1^{1/y}$.

Setup of Public Keys. $\mathcal{A}$ first sets $g_1 = g_1, g_2 = g_2$ and prepares a hash function by $(\kappa, \tau) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g_2, g_2^y)$. Then it chooses $s_1, \ldots, s_q \xleftarrow{\$} \{0,1\}^l$ and aborts if $\mathsf{D}(s_k) = 1_{\mathbb{G}_2}$ for some $k \in [q]$. Next, it sets $k^* \xleftarrow{\$} [q]$, $i^* \xleftarrow{\$} [\ell_1]$, $j^* \xleftarrow{\$} [\ell_2]$. The public key is set as $A_i = g_1^{\tilde{a}_i \prod_{t \in E} \mathsf{d}(t)}$ for all $i \neq i^*$, $A_{i^*} = g_1^{\tilde{a}_{i^*} \prod_{t \in E^*} \mathsf{d}(t)}$, $B_j = g_2^{\mathsf{d}(s_{k^*})\tilde{b}_j}$ for all $j \neq j^*$, and $B_{j^*} = g_2^{\tilde{b}_{j^*}}$ where $\tilde{a}_1, \ldots, \tilde{a}_{\ell_1}, \tilde{b}_1, \ldots, \tilde{b}_{\ell_2} \xleftarrow{\$} \mathbb{Z}_p$. Since $\prod_{t \in E} \mathsf{d}(t) = \prod_{t \in E}(e_t + yf_t)$ and $\prod_{t \in E^*} \mathsf{d}(t) = \prod_{t \in E^*}(e_t + yf_t)$ are polynomials of y with degree at most q, $\mathcal{A}$ can compute all A_i from $(g_1, g_1^y, \ldots, g_1^{y^q})$ even though $\mathcal{A}$ does not explicitly know y. Similarly, $\mathcal{A}$ can compute all B_j from g_2, g_2^y since $\mathsf{d}(s_{k^*}) = e_{s_{k^*}} + yf_{s_{k^*}}$ is a polynomial of y with degree 1. Then, $\mathcal{A}$ gives $PK = (\kappa, g_1, g_2, A_1, \ldots, A_{\ell_1}, B_1, \ldots, B_{\ell_2})$ to $\mathcal{F}$.

Answering Signing Queries. For the k-th signing query, $\mathcal{A}$ has to compute

$$\Big(\prod_{(i,j) \in S(M_k)} g_1^{a_i b_j} \Big)^{1/\mathsf{d}(s_k)} = \prod_{(i,j) \in S(M_k)} \sigma_{i,j}$$

where $\sigma_{i,j} = g_1^{a_i b_j / \mathsf{d}(s_k)}$. We consider two cases. We show $\sigma_{i,j}$ can be computed for all $(i,j) \in M_k$ unless $(i^*, j^*) \in S(M_k) \wedge s_k = s_{k^*}$.

1. If $s_k \neq s_{k^*}$, the $\sigma_{i,j}$ are computed as follows.

$$\sigma_{i,j} = \begin{cases} g_1^{\tilde{a}_i \tilde{b}_j \mathsf{d}(s_{k^*}) \prod_{t \in E_k} \mathsf{d}(t)} = g_1^{\tilde{a}_i \tilde{b}_j (e_{s_{k^*}} + y f_{s_{k^*}}) \prod_{t \in E_k} (e_t + y f_t)} & i \neq i^*, j \neq j^* \\ g_1^{\tilde{a}_i \tilde{b}_j \mathsf{d}(s_{k^*}) \prod_{t \in E_k^*} \mathsf{d}(t)} = g_1^{\tilde{a}_i \tilde{b}_j (e_{s_{k^*}} + y f_{s_{k^*}}) \prod_{t \in E_k^*} (e_t + y f_t)} & i = i^*, j \neq j^* \\ g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E_k} \mathsf{d}(t)} = g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E_k} (e_t + y f_t)} & i \neq i^*, j = j^* \\ g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E_k^*} \mathsf{d}(t)} = g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E_k^*} (e_t + y f_t)} & i = i^*, j = j^* \end{cases}$$

Here, $E_k = E \backslash \{s_k\}$ and $E_k^* = E^* \backslash \{s_k\}$.

2. If $s_k = s_{k^*}$, the $\sigma_{i,j}$ are computed as follows. We assume $(i^*, j^*) \notin S(M_k)$, since otherwise $\mathcal{A}$ aborts.

$$\sigma_{i,j} = \begin{cases} g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E^*} \mathsf{d}(t)} = g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E^*} (e_t + y f_t)} & (i \neq i^*, j = j^*) \vee (i = i^*, j \neq j^*) \\ g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E} \mathsf{d}(t)} = g_1^{\tilde{a}_i \tilde{b}_j \prod_{t \in E} (e_t + y f_t)} & i \neq i^*, j \neq j^* \end{cases}$$

In all these cases, $\sigma_{i,j}$ can be computed efficiently from $g_1, g_1^y, \ldots, g_1^{y^q}$ since $\log_{g_1} \sigma_{i,j}$ is a polynomial of y with degree at most q.

Extracting the Solution to the q-DH Challenge. When $\mathcal{F}$ forges (M^*, σ^*, s^*), $\mathcal{A}$ computes $g_1^{1/y}$ as follows. We assume $s^* = s_{k^*}$ and $(i^*, j^*) \in S(M^*)$, since otherwise $\mathcal{A}$ aborts. If the forgery of the adversary is valid, the following equation holds:

$$\sigma^* = \Big(\prod_{(i,j) \in S(M^*)} g_1^{a_i b_j} \Big)^{1/\mathsf{d}(s^*)} = \prod_{(i,j) \in S(M^*)} \sigma_{i,j}^*.$$

Here we define $\sigma_{i,j}^* = g_1^{a_i b_j / \mathsf{d}(s^*)}$. It is easy to see that, unless $(i,j) \neq (i^*, j^*)$, $\sigma_{i,j}^*$ can be computed efficiently in exactly the same way as in the simulation of the signing oracle. Thus, $\mathcal{A}$ obtains σ_{i^*,j^*}^* by $\sigma^* / \big(\prod_{(i,j) \in S(M^*) \setminus (i^*,j^*)} \sigma_{i,j}^* \big) = \sigma_{i^*,j^*}^*$. Let $\beta(y)$ be a polynomial defined by $\prod_{t \in E^*} (e_t + y f_t) = \sum_{i=0}^{q-1} \beta_i y^i$. We assume that $\beta_0 = \prod_{t \in E^*} e_t \neq 0$ and $e^* = 0, f^* \neq 0$. Otherwise, like the challenger in Game 6, $\mathcal{A}$ aborts. Then $\mathcal{A}$ can compute the solution to the problem by $\Big(\big(\sigma_{i^*,j^*}^*\big)^{f^* / (\tilde{a}_{i^*} \tilde{b}_{j^*})} / \prod_{i=1}^{q-1} (g_1^{y^{i-1}})^{\beta_i} \Big)^{1/\beta_0} = g_1^{1/y}$. □

4.3 Comparison with the Previous Scheme

We compare our scheme with the short signature scheme $\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{cfs}}, \mathsf{H}_{\mathsf{Wat}}]$ in [20]. The differences between our scheme and the other scheme are summarized in Table 1. Two sets of parameters for our scheme are presented in the table. For the first set, $(\ell_1, \ell_2) = (570, 290)$ so that the public key size is the minimum under the condition $\ell_1 \ell_2 = 16mn^2$. With these parameters, the public key size of our scheme is less than 1/100 of that of the other scheme. On the other hand, the computational cost of verification is very high. For the second parameter set, $(\ell_1, \ell_2) = (16530, 10)$. This choice of parameters indicates that we can reduce the size of the public key to about 1/10 of that of the other scheme if we allow a relatively small increase in the computational cost of verification. Signature size and computational cost of signing in our scheme are comparable to those in the other scheme. We also note that the reduction cost of our scheme is essentially the same as that of the other scheme. The only drawback of our scheme compared to the other is a more costly verification algorithm. To sum up, our scheme and $\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{cfs}}, \mathsf{H}_{\mathsf{Wat}}]$ in [20] provide a trade-off between the public key size and the computational cost of verification.

Table 1. Comparison of signature schemes based on the q-DH assumption

Signature scheme	Signature size (bits)	Public key size (bits)	Signing cost	Verification cost
$\mathsf{Sig}_{q\text{-}\mathsf{DH}}[\mathsf{H}_{\mathsf{cfs}}, \mathsf{H}_{\mathsf{Wat}}]$ [20] $(m = 8)$	$\lvert g_1\rvert + \lvert s\rvert$ =200	$16mn^2\lvert g_1\rvert + \lvert s\rvert\lvert g_2\rvert$ $= 2.6 \times 10^7$	$1 \times \mathsf{Exp}$	$2 \times \mathsf{Pairing}$
Ours $(m = 8)$ $(\ell_1, \ell_2) = (570, 290)$	$\lvert g_1\rvert + \lvert s\rvert$ =200	$\ell_1\lvert g_1\rvert + (\lvert s\rvert + \ell_2)\lvert g_2\rvert$ $= 2.0 \times 10^5$	$1 \times \mathsf{Exp}$	$(\ell_2 + 1)\mathsf{Pairing}$ $=291 \times \mathsf{Pairing}$
Ours $(m = 8)$ $(\ell_1, \ell_2) = (16530, 10)$	$\lvert g_1\rvert + \lvert s\rvert$ =200	$\ell_1\lvert g_1\rvert + (\lvert s\rvert + \ell_2)\lvert g_2\rvert$ $= 2.7 \times 10^6$	$1 \times \mathsf{Exp}$	$(\ell_2 + 1)\mathsf{Pairing}$ $=11 \times \mathsf{Pairing}$

The chosen parameters are $\lambda = 80$, $q = 2^{30}, n = 2\lambda = 160$. We also set $l = |s| = \log q + \lambda/m = 40$ so that the term $q^{m+1}/2^{ml}$ in Theorem 2 is at most $1/2^{\lambda}$ as in [20]. We use the BN curve [1] for asymmetric pairing groups and assume that elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ can be represented by $|g_1| = 160$ bits and $|g_1| = 320$ bits, respectively. In our evaluation for computational cost, only the number of exponentiation and pairings are taken into account, and other operations (e.g. individual exponentiations) are ignored.

References

1. Barreto, P.S.L.M., Naehrig, M.: Pairing-Friendly Elliptic Curves of Prime Order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
2. Boneh, D., Boyen, X.: Secure Identity Based Encryption without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
3. Boneh, D., Boyen, X.: Short Signatures without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Boneh, D., Katz, J.: Improved Efficiency for CCA-Secure Cryptosystems Built using Identity-Based Encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
6. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security, pp. 320–329 (2005)
7. Boyen, X., Waters, B.: Shrinking the Keys of Discrete-Log-Type Lossy Trapdoor Functions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 35–52. Springer, Heidelberg (2010)
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. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-Secure Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)
10. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. In: ACM Conference on Computer and Communications Security, pp. 46–51 (1999)

11. Cui, Y., Fujisaki, E., Hanaoka, G., Imai, H., Zhang, R.: Formal security treatments for ibe-to-signature transformation: Relations among security notions. IEICE Transactions 92-A(1), 53–66 (2009)
12. Dodis, Y., Haitner, I., Tentes, A.: On the (in)security of RSA signatures. Cryptology ePrint Archive, Report 2011/087 (2011), http://eprint.iacr.org/
13. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
14. Erdös, P., Frankl, P., Füredi, Z.: Families of finite sets in which no set is covered by the union of two others. J. Comb. Theory, Ser. A 33(2), 158–166 (1982)
15. Fischlin, M.: The Cramer-Shoup Strong-RSA Signature Scheme Revisited. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
16. Gennaro, R., Halevi, S., Rabin, T.: Secure Hash-and-Sign Signatures without the Random Oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
17. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
18. Haralambiev, K., Jager, T., Kiltz, E., Shoup, V.: Simple and Efficient Public-Key Encryption from Computational Diffie-Hellman in the Standard Model. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 1–18. Springer, Heidelberg (2010)
19. Heng, S., Kurosawa, K.: k-Resilient Identity-Based Encryption in the Standard Model. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 67–80. Springer, Heidelberg (2004)
20. Hofheinz, D., Jager, T., Kiltz, E.: Short Signatures from Weaker Assumptions. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 647–666. Springer, Heidelberg (2011), http://eprint.iacr.org/2011/296
21. Hofheinz, D., Kiltz, E.: Programmable Hash Functions and Their Applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
22. Hohenberger, S., Waters, B.: Short and Stateless Signatures from the RSA Assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
23. Katz, J., Vaikuntanathan, V.: Signature Schemes with Bounded Leakage Resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
24. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS (2000)
25. Kumar, R., Rajagopalan, S., Sahai, A.: Coding Constructions for Blacklisting Problems without Computational Assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)
26. Pieprzyk, J., Wang, H., Xing, C.: Multiple-Time Signature Schemes against Adaptive Chosen Message Attacks. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 88–100. Springer, Heidelberg (2004)
27. 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)

28. Waters, B.: Efficient Identity-Based Encryption without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
29. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
30. Yamada, S., Kawai, Y., Hanaoka, G., Kunihiro, N.: Public key encryption schemes from the (B)CDH assumption with better efficiency. IEICE Transactions 93-A(11), 1984–1993 (2010)
31. Zaverucha, G.M., Stinson, D.R.: Short one-time signatures. Cryptology ePrint Archive, Report 2010/446 (2010), http://eprint.iacr.org/

Secure Computation, I/O-Efficient Algorithms and Distributed Signatures

Ivan Damgård, Jonas Kölker, and Tomas Toft*

Dept. of Computer Science, Aarhus University

Abstract. We consider a setting where a set of n players use a set of m servers to store a large, private data set. Later the players decide on functions they want to compute on the data without the servers needing to know which computation is done, while the computation should be secure against a malicious adversary corrupting a constant fraction of the players and servers. Using packed secret sharing, the data can be stored in a compact way but will only be accessible in a block-wise fashion. We explore the possibility of using I/O-efficient algorithms to nevertheless compute on the data as efficiently as if random access was possible. We show that for sorting, priority queues and data mining, this can indeed be done. We show actively secure protocols of complexity within a constant factor of the passively secure solution. As a technical contribution towards this goal, we develop techniques for generating values of form r, g^r for random secret-shared $r \in \mathbb{Z}_q$ and g^r in a group of order q. This costs a constant number of exponentiation per player per value generated, even if less than $n/3$ players are malicious. This can be used for efficient distributed computing of Schnorr signatures. We further develop the technique so we can sign secret data in a distributed fashion at essentially the same cost.

1 Introduction

In this paper, we consider a setting where a set of n players $P_1, \ldots, P_n$ with limited memory use m remote servers $D_1, \ldots, D_m$ to store a large data set securely, and later wish to do secure computation on these data.

As a motivating example, think of a set of authorities in the public sector, each of which initially possesses a database with personal information on various citizens. Suppose they wish to compute results requiring access to all databases, to gain some administrative advantage, for instance, or to do data mining. Allowing a single entity access to everything raises some obvious privacy concerns and is in fact forbidden by law in several countries. A standard solution is to store all data in secret shared form and compute results by multiparty computation.

* The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and from the CFEM research center, supported by the Danish Strategic Research Council.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 278–295, 2012.

But this would require every player to store an amount of data corresponding to all the original databases put together. It may be a more economic and flexible solution to buy storage "in the cloud", i.e., involve some remote servers whose main role is to store the data, and thus we arrive at the situation described above.

An important property of the model is that we do not assume that the computations we want on the database are given in advance. Instead players will decide dynamically which computations to do as a result of input received from the environment – indeed this seems to us to be a more realistic model. This is the main reason why we want the servers to act as storage devices only, rather than have them do computation for us. If the servers had to be involved in the computation, we would have to pay for the communication needed to specify each new computation to all servers. In practice we may also face the problem of installing new software on the servers. In our model, all a server has to do is to supply storage and do a fixed and simple computation on demand, namely whatever is required to read or write a block of data.

We stress that although cloud computing is one motivation for our work, our results can also be applied in a case where no servers are physically present. Namely the players can choose to play the role of the servers themselves. Even in this case, our result offers a way to save memory at little or no extra cost to do the computation. We give more details on this below.

We will assume that our data can be represented as N elements in some finite field, and the computation is specified as a program $\mathcal{P}$ using arithmetic operations in the field, comparison and branching on public data. In addition, the program may read from or write to a "disk", modeling the storage supplied by the servers. Of course, we cannot allow the servers to see any data in cleartext, and since we want to do secure computation on the data later, the computationally most efficient approach is to use secret sharing[1]. A first naive approach is to use standard Shamir secret-sharing[21]; this will require $\Omega(Nm)$ space. A well-known improvement is to use packed secret sharing, suggested by Franklin and Yung [12]. This variant of Shamir's method is a Ramp Scheme[5] that stores data in large blocks (of size $\Theta(m)$ in our case). This will allow us to tolerate a constant fraction of the servers being corrupted, while requiring only storage $O(N)$. However, the usage of packed secret-sharing format means that we can only read or write a large block of data at a time, and we will be wasting resources if we only use a small part of each block we access.

Our first observation is that we can handle this issue by making use of so called I/O-efficient algorithms [1]. In the I/O model, we assume we have a fast computing device (a CPU) with small internal memory, that connects to a large, but much slower memory such as a disk, which only allows you to read or write data in blocks of a given length. The goal of an I/O-efficient algorithm is to solve some computational problem while minimizing the number of times blocks have to be transferred between CPU and external memory. There is a large body

[1] Using fully homomorphic encryption is an option as well, but would incur a huge computational overhead with current state of the art.

of research in this area, and the I/O-complexity of several basic computational problems is known.

Now, if we think of the n players as the CPU and the m servers as the memory device, it is clear that there should be hope that if the players follow an I/O-efficient algorithm, we can minimize the number of transfers and in this way use only $O(N)$ storage while still being able to perform the computation essentially as efficiently as if random access was possible – thus enabling us to use any available technique for optimizing the computation. This idea of combining MPC and I/O-efficient algorithms (which to the best of our knowledge, we are the first to explore) is not completely trivial to capitalize on, however: we need protocols for reading from and writing to the servers' memory. This involves converting between secret sharing among the servers and secret sharing among the players, and must be efficient enough to not dominate the rest of the computation. The program must also be oblivious in the sense that its memory access pattern should only depend on the data known to the adversary, otherwise the construction would clearly be insecure. This, however, is an issue in all known multiparty computation protocols.

In this paper, we give efficient protocols for reading from and writing to the memory formed by the servers, tolerating a constant fraction of malicious players and servers. We build from this a "compiler" that transforms any program that is oblivious (in the sense explained above) into a protocol the runs the program securely in our model. We furthermore show that if the program is also I/O-efficient, then the overhead from the format conversion is insignificant. Here, I/O efficiency means that, up to logarithmic factors, the program needs $O(N/\ell)$ I/O operations, where ℓ is the size of blocks that are transferred. We show that our method applies to a rich class of computational problems by giving oblivious I/O-efficient programs for sorting, priority queues and some forms of datamining.

The table below shows a comparison between running an oblivious and I/O efficient program $\mathcal{P}$ using our approach and "Shamir with servers" which is the naive method where we use standard individual Shamir sharing of all N values on the servers and standard protocols for reading and writing. Here, storage and communication is the total number of field elements while computation is the total number of field operations. $Comp_{\mathcal{P}}(N)$ is the total number of operations done by $\mathcal{P}$ on input size N while $Comm_{\mathcal{P}}(N)$ counts only operations that require communication for a secure implementation, such as multiplication and comparison. We have simplified the table by ignoring logarithmic factors; all details can be found in Section 5.

	storage	communication	computation
Shamir with servers	$O(mN)$	$\Omega(nmN) + Comm_{\mathcal{P}}(N)n$	$\Omega(nmN) + Comp_{\mathcal{P}}(N)n$
Shamir, no servers	$O(nN)$	$Comm_{\mathcal{P}}(N)n$	$Comp_{\mathcal{P}}(N)n$
Our results	$O(N)$	$O(nN) + Comm_{\mathcal{P}}(N)n$	$O(nmN) + Comp_{\mathcal{P}}(N)n$

We see that our technique is in all respects as efficient or better than the "Shamir with servers" approach. The "Shamir, no servers" is a scenario where the players store all the data themselves using standard Shamir sharing and protocols. In this case, the comparison to our results shows that even if players have

massive amounts of memory themselves, our approach still offers a tradeoff between memory usage and computational work: as soon as $Comm_{\mathcal{P}}(N)$ is $\Omega(N)$, our result offers smaller storage, similar or smaller communication complexity, and perhaps a factor m larger computational complexity. We say perhaps because this factor may or may not be significant, depending on how large $Comp_{\mathcal{P}}(N)$ is. For instance, the players could choose to play the role of the servers themselves (so that $m = n$). Then, if $Comp_{\mathcal{P}}(N)$ dominates nN, our results allow saving a factor n memory at no essential extra computation or communication cost.

To achieve our results, we develop some techniques of independent interest: we show how a set of players can compute several values of form $g^r \in G$, where G is a group of prime order, and r is random and secret shared among the players. The cost of this is a constant number of exponentiations per player per instance produced, even if less than $n/3$ players are malicious. To the best of our knowledge, the most efficient solution following from previous work would be for each player to do a Feldman-style VSS [10] of a random value and then combine these shared values to a single one. This would require $\Omega(n)$ exponentiations per player. Our technique can be used, for instance, to do threshold Schnorr signatures very efficiently in an off-line/on-line fashion: by preparing many g^r-values in preprocessing (or in idle time), a signature can be prepared using only cheap linear operations once the message becomes known. While this on-line/off-line property of Schnorr signatures is well-known, it is new that the *total* computational cost per player in a distributed implementation can be essentially the same as what a single player would need to prepare a signature locally. We extend this technique so that the message to be signed can be secret-shared among the players and remains secret, at a constant factor extra cost.

On the technical side, we start from techniques for verifiable secret sharing based on hyperinvertible matrices from [3]. While in [3] information theoretically secure protocols were considered, in our case the secret r is only computationally protected as g^r becomes public. This requires some modification to the protocol but more importantly, some non-trivial issues must be handled in the security proof. We show that our protocol implements an ideal functionality that chooses a random r, secret shares it and publishes g^r. This fixes the value of r, yet the simulator must make a view that is consistent with g^r without knowing r. Finally, we are concerned with the computational efficiency, and a naive extension of the VSS from [3] would cost a factor n more exponentiations than we can afford. To solve this, we devise a technique that allows players to distribute the exponentiations required among themselves while still being able to verify the work of each player efficiently.

2 Preliminaries

As in classical MPC, we have n players $P_1, \ldots, P_n$ who want to compute the value of an arithmetic circuit C over a finite field $\mathbb{F}_q \cong \mathbb{Z}_q$ for a prime q. At most t_p players may be statically corrupted by the adversary, who may be either active or passive. We also have m disk servers $D_1, \ldots, D_m$. They take part in

the protocol, but their role is to store data rather than to compute on it. We assume the adversary can adaptively corrupt at most t_s of these. We assume synchronous communication and secure point to point channels between any pair of players. We will use the UC model for formalizing security of protocols.

We believe that at least our so-called scalable protocols shown towards the end of the paper can be shown secure even against *adaptive* corruption of players. This is on-going work and will be reported on in a future version of the full paper[9].

Recall that Shamir's secret sharing scheme, when sharing one secret among n players, where t of them may be corrupt, uses a polynomial f of degree at most t, and that given $t+1$ points we can reconstruct the polynomial and recompute the secret value $f(0)$. We can think of this as "spending" t points on corrupt players and a single point on storing secrets. If we instead use more points to store secrets and fewer points to handle corrupt players, we can store our data more densely, as observed by Franklin and Yung [12].

Definition 1. *A block is a vector of secrets $(x_1, \ldots, x_\ell)$ over the field $\mathbb{F}$. A block sharing among m servers with threshold t_s is a vector of shares $(f(1), \ldots, f(m))$, where f is a random polynomial of degree at most $d = t_s + \ell - 1 < m$, such that $f(-k) = x_k$ for $k = 1, \ldots, \ell$.*

Given a block $(x_1, \ldots, x_\ell)$ of known values it is easy to generate a suitable polynomial f for a block sharing: let $f(-k) = x_k$ for $k = 1, \ldots, \ell$. Next, choose random values for $f(i)$, for $i = 1, \ldots, t_s + 1$. Use Lagrange interpolation to compute the coefficients of f. Then f can easily be evaluated in $1, \ldots, m$. It is easy to see that any t_s shares reveal no information on the secret block, while any $d+1$ shares allow reconstruction of the entire block. Of course there's nothing special about $-1, \ldots, -\ell$ or $1, \ldots, m$, except the two sets are disjoint and notationally conventient. In the following, we will choose both t_s and ℓ to be linear in m; exact choices will depend, e.g., on whether adversary is passive or not, as detailed later.

In the following, $[x]_f$ will denote a Shamir sharing of value x among the n players using polynomial f, of degree at most t_p. Likewise, for $\mathbf{x} = (x_1, \ldots, x_\ell)$, we let $[[\mathbf{x}]]_g = [[(x_1, \ldots, x_\ell)]]_g$ denote a block sharing among the m servers using polynomial g, of degree at most d as defined above. Depending on the context, we will sometimes omit f and g from the notation. Both types of sharings are linear; in particular, for blocks $\mathbf{x}, \mathbf{y}$ we have $[[\mathbf{x}]]_f + [[\mathbf{y}]]_g = [[\mathbf{x} + \mathbf{y}]]_{f+g}$ where addition of blocks and vectors of shares is done component-wise.

A Word on Broadcast and Byzantine Agreement. When players and servers may be actively corrupted, we sometimes need broadcast among the players for a string of length $\Theta(m)$ field elements. In [11], it is shown how to do this with communication complexity $mn + poly(n)$. Since our protocols communicate at least $O(nm)$ field elements anyway, the cost of this will not dominate if m is sufficiently large compared to n (actually it will always be sufficient if m is cubic in n). However, we emphasize that in practice it will be a much better solution to use protocols without the $poly(n)$ overhead that do not guarantee

termination (see [13]), and take some out-of-band action if the protocol blocks. In this case there is no demand on how m compares to n.

3 The Main Functionality

We will use the UC framework of Canetti [6] to argue security of our protocols. In this framework, one defines what a protocol is supposed to do by specifying an ideal functionality, and then shows that using the protocol is equivalent to using the functionality. The functionality $\mathcal{F}$ that we implement is basically a black-box computer for doing secure arithmetic in $\mathbb{F}$. In addition to these operations, it has a "main memory" and a "disk", and commands for writing to and reading from the disk. A command is executed on request by all honest players.

Functionality $\mathcal{F}$

input(i, v) Get a value x from P_i and store it in main memory at address v.
open(v) Send value stored at location v in main memory to all players and the adversary.
operation$(\diamond, v_1, v_2, v_3)$ Here, $\diamond$ can be $+, -, *$ or $\leq$. Let $val(v_1), val(v_2)$ be the values stored at locations v_1, v_2 in main memory. Compute $val(v_1) \diamond val(v_2)$ (0 or 1 in case of $\leq$), and store the result in location v_3.
const(v, x) Store the constant $x \in \mathbb{Z}_q$ at memory location v.
random(v) Sample a uniformly random $r \in \mathbb{Z}_q$ and store at the memory location v.
write$(addrs, blockaddr)$ Here, *addrs* is a tuple of ℓ distinct addresses in main memory. Write the values stored there as a block on disk, at location *blockaddr*.
read$(addrs, blockaddr)$ As above, *addrs* is a tuple of ℓ distinct addresses in main memory, Read the block from disk at location *blockaddr* and stores the ℓ values obtained in the locations specified in *addrs*.

Fig. 1. The Functionality we implement

4 The Protocols

Standard techniques can be used to implement the input, open and arithmetic (including *const* and *random*) commands of $\mathcal{F}$, so we focus on the read and write commands. These commands reflect that we want the players to store blocks of data on the servers and be able to read the blocks back. As a warm-up we first do this assuming passive corruption, and then show methods for handling malicious players and servers. The write protocol converts ℓ secrets, shared independently, into a block sharing which is given to the servers. The read protocol converts the block sharing back into ℓ separate sharings. For a passive adversary, this is relatively straightforward. The idea is related to the results from [7], except that here we convert between shares held by separate sets of players.

4.1 Passively Secure Implementation of $\mathcal{F}$

We can use the standard technique from [4] to implement the input, output, addition and multiplication commands. This simply amounts to representing a value x stored in main memory by $\mathcal{F}$ as a secret sharing $[x]$. Given this, one can build a (constant round) implementation of the comparison [8].

Hence, to implement the write command, we may assume that the players $P_1, \ldots, P_n$ hold $[x_1], \ldots, [x_\ell]$, that is, sharings of secrets x_1 through x_ℓ to be written. We then implement write by converting this to a block sharing held by the servers. The following lemma is useful

Lemma 1. *For $i = 1, \ldots, m$ and $k = 1, \ldots, d+1$, there exist $\lambda_k^i \in \mathbb{F}$, such that the following holds: For any $x_1, \ldots, x_\ell, r_1, \ldots, r_{d-\ell+1} \in \mathbb{F}$, let f be the polynomial with $f(-i) = x_i, f(j) = r_j$. In other words, f defines a block-sharing $[[(x_1, \ldots, x_\ell)]]_f$. Then each share in this block sharing can be computed as a linear combination of the x_i's and r_j's. More precisely,*

$$f(i) = \sum_{k=1}^{\ell} \lambda_k^i x_k + \sum_{k=\ell+1}^{d+1} \lambda_k^i r_{k-\ell}$$

Proof. If we set $f(-i) = x_i$ and $f(j) = r_j$ (for all sensible i and j), this uniquely defines a polynomial f of degree at most d. This can be computed through Lagrange interpolation, which is a linear computation. More precisely, Let $y = (x_1, \ldots, x_\ell, r_1, \ldots, r_{d-\ell+1})^\top$. Writing $f(x) = c_0 x^0 + \ldots + c_d x^d$, let $c = (c_0, \ldots, c_d)^\top$, and let V be a $(d+1) \times (d+1)$ Vandermonde matrix over the points $-\ell, \ldots, -1, 1, \ldots, d-\ell+1$. Then $V \cdot c = y$ and thus $V^{-1} \cdot y = c$. We can then compute $f(x)$ as $(x^0, \ldots, x^d) \cdot c$. Since each entry in c is a linear combination of entries in y, so is $(x^0, \ldots, x^d) \cdot c$ and this defines the λ_k^i's we promised.

In the protocol we assumes that players can generate shares of random values unknown to the adversary. Several techniques exist for this, and for now, we simply assume access to a functionality F_{Rand}, defined below, and discuss later how to implement it.

Functionality F_{Rand}

Share(*num*) For $i = 1, \ldots, num$, choose random $s_i \in \mathbb{F}$, form $[s_i]_{f_i}$, where f_i is random subject to $f_i(0) = s_i$, $deg(f_i) \leq t_p$, and in each point owned by a corrupt player, f_i evaluates to a share chosen by the adversary[a]. Finally, send shares of all *num* values to all honest players.

[a] We need to allow the adversary to choose shares for corrupted players, in order to be able to implement the functionality.

Fig. 2. The functionality delivering randomness

The idea behind the definition of F_{Rand} is that we can use known techniques to implement it for large values of the *num* parameter, with low amortized cost per sharing generated.

Protocol $Write(addrs, blockaddr)$

1. Call $Share(d - \ell + 1)$ to get sharings of random values $[r_1], \ldots, [r_{d-\ell+1}]$.
2. For $i = 1, \ldots, m$ compute

$$[f(i)] = \sum_{k=1}^{\ell} \lambda_k^i [x_k] + \sum_{k=\ell+1}^{d+1} \lambda_k^i [r_{k-\ell}]$$

   where $\{[x_k]\}$ is a set of individually shared values pointed to by *addrs* and f is the polynomial from Lemma 1. Note that this only requires local computation.
3. For $i = 1, \ldots, m$ each player sends "write *blockaddr*" and his share of $[f(i)]$ to server D_i.
4. For $i = 1, \ldots, m$ D_i uses the shares received to reconstruct and store $f(i)$ under *blockaddr*.

Fig. 3. Protocol for writing a block

Next, let us consider a passively secure protocol for reading a block. Each server i has a share from the sharing of the block to be read, $[(x_1, \ldots, x_\ell)]$. We implement read by converting this to individual sharings held by the players, $[x_1], \ldots, [x_\ell]$. For this, we use the following lemma which is easy to show in a way similar to Lemma 1:

Lemma 2. *There exist constants* δ_j^i *such that for any block sharing* $[x_1, \ldots, x_\ell]_f$, *we have* $x_i = \sum_{j=1}^{m} \delta_j^i f(j)$.

Protocol $Read(addrs, blockaddr)$

1. each player sends "read *blockaddr*" to each server.
2. Each server D_i retrieves $f(i)$ using *blockaddr*, forms a set of shares $[f(i)]$ from $f(i)$ and sends a share to each player.
3. For $k = 1, \ldots, \ell$, the players locally compute $[x_k] = \sum_{i=1}^{m} \delta_i^k [f(i)]$ and associate the results with the addresses in *addrs*.

Fig. 4. Protocol for reading a block.

Security of the read and write protocols follow from a standard and straightforward simulation argument, which can be found in the full paper[9].

By inspection, it is easy to see that each call to *Read* or *Write* involves communication of $O(nm)$ field elements and $O(nm^2)$ local field operations.

4.2 Implementation for Malicious Servers and Players

In this section we show how to handle the case where a constant fraction of the servers and players are corrupted by a malicious adversary. A first easy observation is that when less than $t_p < \frac{n}{3}$ players are actively corrupted, we can implement the input, open and arithmetic operations of $\mathcal{F}$ using standard techniques from [4] such that ordinary Shamir secret sharing is still the way values are represented. Moreover, when an honest server receives shares of some value from the players, it can use standard error correction techniques to reconstruct the right value. We need, of course, an implementation of $\mathcal{F}_{Rand}$ with active security and we show how to do this later.

The main problem is that malicious servers may return incorrect shares in read operations. While an obvious solution is to authenticate each share held by a server, this is not trivial to do efficiently: since shares must be kept private, the players have to do a secure computation of the authentication value. We present two solutions that work along these lines. The first uses information theoretically secure macs, leading to protocols *RobustRead* and *RobustWrite* found in the full paper[9]. We get the following result:

Theorem 1. *Together with the standard techniques for implementing the input, open and arithmetic operations, RobustRead and RobustWrite form a statistically secure implementation of $\mathcal{F}$ in the F_{Rand}-hybrid model for an adversary corrupting at most $t_p < n/4$ players and $t_s \in \Theta(m)$ servers actively, or an adversary corrupting at most $t_p < n/2$ players passively and $t_s \in \Theta(m)$ servers actively. Each call to RobustRead or RobustWrite involves communication of $O(nm)$ field elements and $O(nm^2)$ local field operations.*

This solution is relatively simple but needs extra functionality from F_{Rand} that we only know how to implement for passive corruption or for a small number of players in case of active corruption. In the next section, we present a second solution that scales better with the number of players.

Additional functions for F_{Rand}

$share^{+}(num)$ Same as *share*, generate *num* random shared values $[r_i]$, but in addition send α^{r_i} to all players.
$share^{++}(num)$ Generate *num* pairs of random shared values $[u_i], [v_i]$ in the same way as in *share*, but in addition, send $g^{u_i}h^{v_i}$ to all players.

Fig. 5. Additional randomness functions for the scalable solution

4.3 A Scalable Method for Handling Malicious Players and Servers

In this section, we show how to handle actively corrupted players and servers in a way that scales well with n. The solution is based on making Schnorr signatures [20] on Pedersen commitments [18] to a combination of the data and a sequence number (which prevents replay attacks). The Schnorr signatures are

in a group G of order q where q is the size of $\mathbb{F}$, the field we compute in, and where we assume q to be prime. We will define protocols $ScalableRobustWrite$ and $ScalableRobustRead$ and show the following:

Theorem 2. *Assuming that Pedersen commitments and Schnorr signatures in G are secure, then, together with the standard techniques for implementing then, together with the standard techniques for implementing the input, open and arithmetic operations, the protocols $ScalableRobustRead$ and $ScalableRobustWrite$ form a computationally secure implementation of $\mathcal{F}$ in the F_{Rand}-hybrid model for an adversary corrupting at most $t_p < n/3$ players and $t_s \in \Theta(m)$ servers actively. $ScalableRobustRead$ and $ScalableRobustWrite$ each involve communication of $O(nm)$ field elements and $O(nm^2)$ local field operations plus $O(nm)$ exponentiations in G per invocation.*

The $O(nm)$ exponentiations in G we require are equivalent to $O(\log(|\mathbb{F}|)nm)$ multiplications in G. To see if this extra cost is going to dominate, consider that if we use a state of the art implementation of G based on elliptic curves, a multiplication in G takes time comparable to a multiplication in $\mathbb{F}$. So comparing $O(nm^2)$ field operations to $O(\log(|\mathbb{F}|)nm)$ multiplications in G boils down to comparing m and $\log(|\mathbb{F}|)$. These parameters are not really comparable in general, but since the idea of our model is to consider a moderate number of players and many servers, it does not seem unreasonable to consider m and $\log(|\mathbb{F}|)$ to be the same order of magnitude. Under this assumption, the solution has the same complexity as the passively secure protocol up to a constant factor. We add that in many settings, the communication cost is the main limiting factor, in which case our solution will be satisfactory regardless of other factors.

Some words on the basic ideas of our protocols: the main problem we face is that the servers may send incorrect data in the read protocol. To protect against this, the players will sign each server's share using a shared signing key. However, this share must not become publicly known and we know of no signature scheme where we can sign efficiently when both the signing key and the message are secret-shared. To overcome this, we instead sign a commitment to the server's share since this commitment can indeed be publicly known. We also have the problem of the servers replaying old signed values, but this can be solved easily by maintaining a sequence number $c_{blockaddr}$ (initally 0) counting how many times we wrote to *blockaddr*, signing this, and having the servers remember it.

Next we describe the protocol in more detail. First, some setup: let G be a group of order q where random elements $g, h \in G$ have been chosen to be used as public key for Pedersen commitments in G, that is, a commitment to $m \in \mathbb{Z}_q$ is of form $g^m h^r$ for random $r \in \mathbb{Z}_q$. We also assume that a verification key α, β for our Schnorr signature scheme has been set up, with $\beta = \alpha^a$ for some $a \in \mathbb{Z}_q$ where the players hold a sharing $[a]$ of a. This is formalized in the UC model by assuming a functionality that outputs these values initially.

Recall that in Schnorr's scheme, a signature on m is a pair (γ, δ) satisfying $\gamma = \alpha^\delta \cdot \beta^{H(\gamma,m)}$ for some collision intractable hash function H. Observe that signatures can be easily verified, and computed knowing a by setting $\gamma = \alpha^r$ and

Protocol $ScalableRobustWrite(addrs, blockaddr)$

1. Send "begin write at *blockaddr*" to the servers. Each server returns its value of $c_{blockaddr}$, players decide the correct value by majority, and increment the value.
2. Call $Share(d-\ell+1)$ to get sharings of random values $[r_1], \ldots, [r_{d-\ell+1}]$.
3. For $i = 1, \ldots, m$ compute

$$[f(i)] = \sum_{k=1}^{\ell} \lambda_k^i [x_k] + \sum_{k=\ell+1}^{d+1} \lambda_k^i [r_{k-\ell}]$$

   where $\{[x_k]\}$ are a set of individually shared values pointed to by $addrs$ and f is the polynomial from Lemma 1. Note that this only requires local computation. In the following, we set $s_i = f(i)$.
4. Call subprotocol $share^{++}(m)$ to obtain $c'_i = g^{u_i} h^{v_i}$ and sharings $[u_i], [v_i]$ for $i = 1 \ldots m$. Also call $share(1)$ to get $[x]$.
5. We now want to adjust c'_i so it becomes a commitment to s_i. Players compute locally shares in $[s_i - u_i]$ and send them to P_u who computes $\tau_i = s_i - u_i$ and broadcasts all the τ_i's. To verify P_u's work, players open x by sending all shares to everyone and compute locally $\sum_i x^i([s_i] - [u_i] - \tau_i) = [\sum_i x^i(s_i - u_i - \tau_i)]$. This value is opened by each player sending his share to all players. If the resulting value is 0 continue to the next step. Otherwise, P_u is disqualified, we set $u = u+1$, and all players exchange shares of $s_i - u_i$ and compute the correct values τ_i.
6. All players compute $c_i = g^{\tau_i} c'_i$. We now want to sign c_i (and some more data).
7. Call $share^{+}(m)$ to obtain a sharing $[r_i]$ of some random value $r_i \in \mathbb{Z}_q$, and $\gamma_i = \alpha^{r_i}$, for $i = 1 \ldots m$.
8. Using hash function H, each player computes their share $[\delta_i]_j = [r_i] - [a]H(\gamma_i, c_i, c_{blockaddr})$ of δ_i.
9. The players all send "complete write at *blockaddr* using $([s_i], [v_i], [\delta_i], \gamma_i)$" to server i.
10. Each server i reconstructs s_i, v_i, δ_i using error correction to handle incorrect shares from corrupt players, computes γ_i (taking majority decision on the values received), increments $c_{blockaddr}$ and stores all values.

Fig. 6. Scalable protocol for writing a block

$\delta = r - a \cdot H(\gamma, m)$ for a random $r \in \mathbb{Z}_q$. We will need functions from F_{Rand} in addition to those we defined earlier. The new functions are defined in Figure 5.

Our protocol as well as some subprotocols described later uses player elimination where some players are disqualified underway, so that only a subset of the players actually participate at any given time. For ease of exposition, we suppress this fact in our description below, but emphasize that an actual implementation must keep track of who participates. Furthermore, in the protocol below for reading and writing, we use a global variable u that is remembered between calls to the read/write protocols. It points to a player and is initially 1. P_u is a player that does computational work on behalf of all players, to save resources. If P_u is found to be corrupt, he is replaced by P_{u+1}. In such a case,

Protocol *ScalableRobustRead*$(addrs, blockaddr)$

1. All players send "read at *blockaddr* towards P_u" to all the servers. The servers each compute u by majority decision.
2. Each server D_i sends γ_i, δ_i, c_i to player u and a share from $[s'_i], [v'_i]$ to each player, with the implied claim that $(s'_i, v'_i) = (s_i, v_i)$ as last stored by the read protocol, and that $c_i = g^{s_i} h^{v_i}$. The current value of $c_{blockaddr}$ is also sent, and players decide on reception on the correct value of $c_{blockaddr}$ by majority.
3. Since we cannot rely on servers to supply consistent shares of s'_i, v'_i, players generate their own consistent shares and adjust them to what the (honest) servers sent: Call $share^{++}(m)$ to obtain sharings $[b_{x,i}]$ and $[b_{y,i}]$ of random elements of $\mathbb{F}$, for $i = 1, \ldots, m$ as well as a commitment $g^{b_{x,i}} h^{b_{y,i}}$ to $b_{x,i}$, using $b_{y,i}$ as the randomness.
4. For $i = 1, \ldots, m$, everybody sends their shares in $[s'_i - b_{x,i}]$ and $[v'_i - b_{y,i}]$ to P_u. P_u attempts to reconstruct $x_i = s'_i - b_{x,i}$ and $y_i = v'_i - b_{y,i}$. He verifies for all i that (γ_i, δ_i) is a valid signature on $c_i, c_{blockaddr}$ and that $c_i = g^{x_i} h^{y_i} \cdot g^{b_{x,i}} h^{b_{y,i}} (= g^{x_i + b_{x,i}} h^{y_i + b_{y,i}} = g^{s'_i} h^{v'_i})$. This may fail for some i's, in which case P_u sets $x_i = y_i = \perp$.
5. P_u broadcasts γ_i, δ_i, c_i and (x_i, y_i), for $i = 1, \ldots, m$ to the players with the implied claim that $x_i = s'_i - b_{x,i}$ and $y_i = v'_i - b_{y,i}$.
6. Each player verifies for all i that (γ_i, δ_i) is a valid signature on $c_i, c_{blockaddr}$ and that $c_i = g^{x_i} h^{y_i} \cdot g^{b_{x,i}} h^{b_{y,i}}$. Let S_u be the set of i for which this holds. If S_u is smaller than $m - t_s$, disqualify P_u, set $u = u + 1$ and restart the read protocol. Otherwise, the players compute $[b_{x,i}] + x_i = [b_{x,i} + x_i]$ for $i \in S_u$. Note that unless the signature scheme or commitment scheme has been broken, $b_{x,i} + x_i = s'_i = s_i$.
7. For $k = 1, \ldots, \ell$, the players compute locally

$$[x'_k] = \sum_{i=1}^{m} \delta_i'^k [b_{x,i} + x_i]$$

and associates the results with the addresses in *addrs*. Here $\delta_i'^k$ is a set of interpolation coefficients such that $\delta_i'^k = 0$ if i is not in S_u, and the rest is set such that the correct value for x'_k is computed. This is possible because we set the parameters such that there are enough honest servers to reconstruct only from their data. This still allows t_s to be in $\Theta(m)$.

Fig. 7. Scalable protocol for reading a block

our protocols usually do an amount of work that is much larger than in normal operation, but since this can only happen t_p times, it only incurs an additive overhead that is independent of the size of the computation we do. Therefore it does not affect the asymptotic complexity of our solution.

We assume that Pedersen commitments and Schnorr signatures are secure. Pedersen commitments are perfectly hiding, and computationally binding assuming that computing the discrete log of h base g is hard. We assume Schnorr signatures to be secure against existential forgery under chosen plaintext attacks, which they are in the random oracle model as long as computing the discrete

log of β base α is hard. If the sequence numbers are public knowledge (i.e. known to the adversary and beyond his control), security only against known plaintext attacks might be sufficient: in that case the messages are a combination of something uniformly random (a Pedersen commitment) and something known in advance. This might require the computed algorithm (or at least its I/O behavior) to be oblivious.

A more detailed argument for security of these protocols can be found in the full paper [9]. The most important observation in the write protocol is that in step 5, players effectively evaluate a polynomial with coefficients $s_i - u_i - \tau_i$ in a random point x. If there is an error, i.e., $\tau_i \neq s_i - u_i$ for some i, the polynomial is not zero and can have at most m roots. Hence the check is OK with probability at most $m/|\mathbb{F}|$ which is negligible. Note also that in the read protocol, an honest P_u will always get good data from the honest servers and hence can make S_u be of size at least $m - t_s$—this also means P_u must be corrupt if S_u is too small. On the other hand, even if P_u is corrupt, the commitments and signatures are checked by everyone and hence the last step is always successful if S_u is large enough. The resulting protocols are shown in Figure 6 and Figure 7.

4.4 Implementation of F_{Rand}

To implement the *Share* command, we can use a protocol from [3], which is based on so-called hyperinvertible matrices. A matrix is hyperinvertible if any square submatrix is invertible.

We sketch the idea here and refer to [3] for details. Using an n by n hyperinvertible matrix M, players can generate $\Theta(n)$ random shared values at cost $O(n)$ communication per value: each player P_i simply constructs $[r_i]$ for random r_i and sends shares to the other players. Now each player P_j collects a vector $(r_{1,j}, r_{2,j}, \ldots, r_{n,j})$ of shares he received, multiplies the vector by M to obtain a new vector $(s_{1,j}, s_{2,j}, \ldots, s_{n,j})$, and outputs $(s_{1,j}, s_{2,j}, \ldots, s_{n-t_p,j})$. Clearly this forms shares of the first $n - t_p$ entries in $M \cdot (r_1, \ldots, r_n)$. By the hyperinvertible property, these entries are in 1-1 correspondence with the $n - t_p$ values chosen by honest players, and are therefore completely unknown to the adversary. We get the required efficiency, as long as t_p is any constant fraction of n, and [3] shows how to add checks to get security also for $t_p < n/3$ actively corrupted players.

In Figure 8 we show a protocol for the $share^+$ command that produces $\Theta(nm)$ values of form $[r], \alpha^r$ where the r's are randomly chosen and the only information released to the adversary on r is α^r. Figure 9 shows a subprotocol required to compute the results in step 3 of $share^+$ efficiently.

A protocol for the $share^{++}$ command is obtained by a straightforward extension of the $share^+$ protocol and can be found in the full paper [9].

As for security of the protocols, correctness of the checks in steps 2e and 3 of $share^+$ can be argued in a way similar to the check in the *ScalableRobustRead* protocol. Security then follows from this and the hyper-invertible property of M. See the full paper [9] for a simulation proof.

Protocol $share^+$:

1. Use the protocol from [3] to generate random shared values $[r_b^a]$ and $[x_a]$ for $a = 1..n$, $b = 0..m$.
2. In parallel, for $a = 1, \ldots, n$ do:
   (a) Players send their shares in $[r_b^a]$ to P_a, for $b = 0 \ldots m$.
   (b) P_a reconstructs the r_b^a's, computes $\chi_b^a = \alpha^{r_b^a}$ and broadcasts these values, for $b = 0 \ldots m$.
   (c) Now we want to verify that P_a has computed correctly, so we open x_a by broadcasting all its shares.
   (d) Locally compute $[y_a] = [\sum_{b=0}^m {x_a}^b r_b^a] = [r_0^a] + [\sum_{b=1}^m {x_a}^b r_b^a]$ and open y_a to everyone.
   (e) All players check that $\alpha^{y_a} = \prod_{b=0}^m (\chi_b^a)^{(x_a)^b}$. If this is not satisfied, P_a is disqualified, open r_b^a for $b = 0, \ldots, m$, and all players compute the correct values of $\alpha^{r_b^a}$. Hence, in any case, we can assume we continue with correct values of $\chi_b^a = \alpha^{r_b^a}$, for $a = 1 \ldots n$, $b = 0 \ldots m$.
3. We form column vectors $V_b, b = 1 \ldots m$, with n entries, where entry a is of form $([r_b^a], \chi_b^a)$. Players then compute a new column vector $M \cdot V_b$, formed as follows: let $\gamma_1, \ldots, \gamma_n$ be the k'th row of M. The k'th entry of $M \cdot V_b$ is $([\sum_a \gamma_a r_b^a], \prod_a (\chi_b^a)^{\gamma_a} = \alpha^{\sum_a \gamma_a r_b^a})$. To get the required efficiency, it is necessary that the players share the work of doing the exponentiations. We do this using subprotocol *AmortizedExp* described below.
4. Output the first $n - t_p$ entries of $M \cdot V_b$, for $b = 1 \ldots m$.

Fig. 8. Protocol for sharing random exponents

Protocol *AmortizedExp*:

1. Each player P_k computes a part of the result, namely $\beta_b^k = \prod_{a=1}^n (\chi_b^a)^{\gamma_a}$, for $b = 1 \ldots m$, where $(\gamma_1, \ldots, \gamma_n)$ is the k'th row of M. P_k broadcasts his results.
2. Generate and open a random value x in the same way as in $share^+$.
3. All players compute $(\delta_1, \ldots, \delta_n) = (x^0, \ldots, x^{n-1}) \cdot M$, i.e., we take a linear combination of the rows of M. Players check that, for $b = 1, \ldots, m$, we have:

$$\prod_{k=1}^n (\beta_b^k)^{x^{k-1}} = \prod_{a=1}^n (\chi_b^a)^{\delta_a}$$

   If this does not hold, at least one player output incorrect results. All players then compute all the β_b^k, find the player(s) who cheated and disqualify them. In any case, output the β_b^k's.

Fig. 9. Protocol for exponentiating many values

5 Running Oblivious Algorithms on $\mathcal{F}$

So far we've discussed how to implement a "black box" functionality called $\mathcal{F}$. In this section, we show how to execute oblivious programs that use $\mathcal{F}$ as a library of subroutines, and analyze the time and communication cost of doing so.

First, we need to define what a program for $\mathcal{F}$ is. We shall take it to mean a finite sequence of instructions $\mathcal{C}$, each of which is either one of $\mathcal{F}$'s commands with suitable parameters, or a control command $(\texttt{goto}, i)$ which is described below.

These programs are executed on a machine with a memory $\mathcal{M}$, the configurations of which are $\in \mathbb{Z}_q^M$. The machine also has a program counter register, $pc \in \mathbb{Z}_q$ and a finite disk $\mathcal{D}$ with configurations in $(\mathbb{Z}_q^\ell)^D$ where ℓ is the block size. All $\mathbb{Z}_q$-elements are initially set to 0. We assume that $q \geq M, D, (|\mathcal{C}| + 1)$[2].

To execute a program, repeat this main loop: Leak pc. If $pc \geq |\mathcal{C}|$, halt; else if $\mathcal{C}[pc] = (\texttt{goto}, v)$, set $pc \leftarrow \mathcal{M}[v]$; else act according to $\mathcal{C}[pc]$ as described by $\mathcal{F}$ and subsequently increment pc. Then repeat.

With these instructions, a fixed-address jump instruction "goto p" can be expressed as the sequence $(\texttt{const}, 0, p)$, $(\texttt{goto}, 0)$. A conditional jump "if $\mathcal{M}[v_1] \diamond \mathcal{M}[v_2]$ goto x else goto y" where $\diamond$ is one of the two comparisons, can be expressed as the sequence $(\diamond, 0, v_1, v_2)$, $(\texttt{const}, 1, x-y)$, $(*, 0, 1, 0)$, $(\texttt{const}, 1, y)$, $(+, 0, 1, 0)$, $(\texttt{goto}, 0)$. Out of these, control structures like **if-then-else**, **while** and **for** can be built.

A program $\mathcal{P}$ that makes random choices can be viewed as a function mapping inputs to distributions of streams of leakage. A program is said to be *perfectly oblivious* if the output reveals this distribution for each input that can produce that output. Formally speaking, $\mathcal{P}$ is oblivious if there exists a polynomial time randomized turing machine T such that

$$\forall o \forall i \colon (\exists r \colon \mathcal{P}(i, r) = o) \Rightarrow T_{\mathcal{P}}(o) \sim^p traces(\mathcal{P}, i).$$

That is to say, for each output o and for each input i that can make $\mathcal{P}$ produce o as the output, the distribution of instructions executed by $\mathcal{P}$ (over $\mathcal{P}$'s coin flips r) can be sampled by T using just the output. Computational and statistical obliviousness is defined similarly, where T samples suitably indistinguishable distributions. If a program is perfectly oblivious, we say it is *oblivious*.

A note about this definition: a more commonly used definition of obliviousness is that the distribution of traces is computable knowing only the program and not the output. Here, what we want to capture is really that the program is oblivious from the adversary's point of view; that is to say, given what the adversary is allowed to know in the ideal world (the output), he could himself have computed what he sees in the real world. Using this definition, we are now ready to state the following theorem:

Theorem 3. *Given an oblivious program $\mathcal{P}$ for $\mathcal{F}$, we can implement using $\mathcal{F}$ a UC-secure functionality $\mathcal{F}_{\mathcal{P}}$ that has only* `input` *and* `output` *commands, such*

[2] Otherwise, we could express things in terms of an implicit "bignum" library, but this would be unwieldy.

that if a set of inputs is given to $\mathcal{P}$ and the same inputs are given to $\mathcal{F}_{\mathcal{P}}$, the outputs produced by $\mathcal{F}_{\mathcal{P}}$ equal those produced by $\mathcal{P}$ when run on $\mathcal{F}$.

Proof. To implement $\mathcal{F}_{\mathcal{P}}$, simply run $\mathcal{P}$ on $\mathcal{F}$ as described above. This clearly creates the desired relationship between inputs and outputs. A UC simulator takes the output from running $\mathcal{F}$ on $\mathcal{P}$ and feeds it to the turing machine T that exists because $\mathcal{P}$ is oblivious. This yields a simulated trace of $\mathcal{P}$ that is suitably (i.e. perfectly, statistically or computationally) indistinguishable from the real world.

In the full paper[9], we show a detailed analysis of the efficiency of the protocol we obtain by using the above theorem for an i/o efficient program $\mathcal{P}$ together with our scalable implementation of $\mathcal{F}$. This analysis leads to the table found in the introduction.

6 Example Applications

There are plenty of example applications for the previous theorem. Oblivious algorithms have already been studied in literature; sorting networks are the prime example. Another source of examples is MPC protocols for specific tasks. Oblivious algorithms are often used as the basis for protocols, where two or more parties wish to query or process some secret dataset. The present setting is very suited for such examples: The data is often processed during one or more scans of the entire dataset.

Secure Datamining. Secure datamining was introduced by Lindell and Pinkas [17] and k-means clustering is one example. Given N points p_j in a d-dimensional space, group these points into k *clusters*, i.e., find trends in the data. The k-means clustering protocol of Jagannathan and Wright [16] is easily generalized to consider a secret shared database held by a number of disk-servers.

The overall idea is to start with a set of k initial *cluster centers*. These are then updated: 1) assign each p_j to the closest center, 2) replace each center with the average of its points. The process is repeated until some (revealed) termination condition is fulfilled, e.g., the clusters move sufficiently little. The details are seen below; the d-dimensional points are stored as d secret shared coordinates. To improve readability, we refer to shared points as a single secret sharing, writing $[p_j]$ rather than $([p_{j,1}], \ldots, [p_{j,d}])$.

- Init cluster centers, $[c_i]$
- `While` Termination condition not reached `do`
  - Init accumulators ($[N_i]$: number of points assigned to $[c_i]$; $[\sigma_i]$: sum of points assigned to $[c_i]$)
  - `For` each point $[p_j]$
    - Determine index, $[k]$, of closest cluster
    - `For` each cluster center $[c_i]$: Add $([k] =^? i)$ to $[N_i]$ and $([k] =^? i) \cdot [p_i]$ to $[\sigma_i]$
  - `For` each cluster center $[c_i]$: $[c_i] \leftarrow [\sigma_i] \div [N_i]$

We ignore many details, notably computing distances and averages, and determining the closest center. Our goal is *not* to provide a k-means clustering protocol, but to demonstrate that the task can be performed with most of the data residing "on disk.". Access is oblivious and I/O-efficient: the parties hold the $[c_i]$, $[\sigma_i]$, and $[N_i]$ in memory, while the dataset (the $[p_j]$) is accessed by linear scans.

Sorting. As a second example, consider a Shell sorting (SS) network, sorting N elements: for each value $\delta = 2^j 3^k < N$ in decreasing order, for $i = 1, \ldots, N - \delta$, compare and exchange elements i and $i + \delta$. Clearly that's $O(N \log_2 N \log_3 N)$ comparisons, and they're organised in $O(\log^2 N)$ pairs of parallel forward scans. So SS does $IO_{SS}(N) = \frac{N}{\ell} \log^2 N$ I/O, and $Comp_{SS}$ is $\Theta(N \log^2 N)$: it falls in the class of I/O-efficient oblivious programs we analyzed in the previous section. For proof of correctness, see [19].

There are many alternatives to the SS network. The odd-even merge sort [2] provides the same result, but better round complexity. Randomized SS [14] seems to require many I/O operations as it (recursively) compares elements in a randomized complete matching between the left and right halves of the data. Using an I/O-efficient oblivious sorting algorithm (e.g., the odd-even merge sort adaptation in [15]) appears to be a good strategy, but even if we can get to $O(\frac{N}{\ell} \log_{\frac{M}{\ell}} \frac{N}{\ell})$ I/O ([15] gets to $O(\frac{N}{\ell} \log^2_{\frac{M}{\ell}} \frac{N}{\ell})$), we still need $\Omega(N \log N)$ comparisons, each of which requires computation and communication.

A priority queue. Toft's oblivious priority queue (PQ) [22] is an oblivious datastructure allowing two operations: `Insert` $([x], [p])$ which adds element $[x]$ with priority $[p]$; and `GetMin` $()$, which removes and returns the element with minimal priority. The solution keeps track of a list that is sorted lazily, through buffering. Extraction of the minimal pair is trivial, except for occasional buffer flushing.

The construction is based entirely on oblivious merging as in Batcher's odd-even merge sort [2], which is I/O-efficient. Combining the techniques of this paper with those of [22] implies a PQ with the bulk of the data held on disk. This makes perfect sense: the bulk of the data is very rarely touched, hence storing it in a compact fashion ensures a very low memory overhead with little impact on efficiency.

References

1. Aggarwal, A., Vitter, S., Jeffrey: The input/output complexity of sorting and related problems. Commun. ACM 31, 1116–1127 (1988)
2. Batcher, K.E.: Sorting networks and their applications. In: Proceedings of the Spring Joint Computer Conference, AFIPS 1968, April 30-May 2, pp. 307–314. ACM, New York (1968)
3. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-Secure MPC with Linear Communication Complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008)

4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 1–10. ACM, New York (1988)
5. Blakley, G.R., Meadows, C.: Security of Ramp Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 242–268. Springer, Heidelberg (1985)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, (2000), http://eprint.iacr.org/
7. Cramer, R., Damgård, I.B., de Haan, R.: Atomic Secure Multi-Party Multiplication with Low Communication. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 329–346. Springer, Heidelberg (2007)
8. Damgård, I.B., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally Secure Constant-Rounds Multi-Party Computation for Equality, Comparison, Bits and Exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006)
9. Damgård, I., Kölker, J., Toft, T.: Secure computation, i/o-efficient algorithms and distributed signatures. Cryptology ePrint Archive (2011), http://eprint.iacr.org/
10. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS, pp. 427–437 (1987)
11. Fitzi, M., Hirt, M.: Optimally efficient multi-valued byzantine agreement. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, pp. 163–168. ACM, New York (2006)
12. Franklin, M., Yung, M.: Communication complexity of secure computation (extended abstract). In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing, STOC 1992, pp. 699–710. ACM, New York (1992)
13. Goldwasser, S., Lindell, Y.: Secure multi-party computation without agreement. J. Cryptology 18(3), 247–287 (2005)
14. Goodrich, M.T.: Randomized shellsort: A simple oblivious sorting algorithm. In: SODA, pp. 1262–1277 (2010)
15. Goodrich, M.T., Mitzenmacher, M.: Mapreduce parallel cuckoo hashing and oblivious ram simulations. CoRR, abs/1007.1259 (2010)
16. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD 2005, pp. 593–599. ACM, New York (2005)
17. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000)
18. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
19. Pratt, V.R.: Shellsort and Sorting Networks. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York (1972), http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/shell/shellen.htm
20. Schnorr, C.-P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
21. Shamir, A.: How to share a secret. Commun. ACM 22, 612–613 (1979)
22. Toft, T.: Secure datastructures based on multiparty computation. Cryptology ePrint Archive, Report 2011/081 (2011), http://eprint.iacr.org/

Delegatable Homomorphic Encryption with Applications to Secure Outsourcing of Computation

Manuel Barbosa[1] and Pooya Farshim[2]

[1] HASLab/INESC TEC, Universidade do Minho, Braga, Portugal
[2] Department of Computer Science, Darmstadt University of Technology, Germany
mbb@di.uminho.pt, farshim@cased.de

Abstract. We propose a new cryptographic primitive called Delegatable Homomorphic Encryption (DHE). This allows a Trusted Authority to control/delegate the evaluation of circuits over encrypted data to untrusted workers/evaluators by issuing tokens. This primitive can be both seen as a public-key counterpart to Verifiable Computation, where input generation and output verification are performed by different entities, or as a generalisation of Fully Homomorphic Encryption enabling control over computations on encrypted data. Our primitive comes with a series of extra features: 1) there is a one-time setup procedure for all circuits; 2) senders do not need to be aware of the functions which will be evaluated on the encrypted data, nor do they need to register keys; 3) tokens are independent of senders and receiver; and 4) receivers are able to verify the correctness of computation given short auxiliary information on the input data and the function, independently of the complexity of the computed circuit. We give a modular construction of such a DHE scheme from three components: Fully Homomorphic Encryption (FHE), Functional Encryption (FE), and a (customised) MAC. As a stepping stone, we first define Verifiable Functional Encryption (VFE), and then show how one can build a secure DHE scheme from a VFE and an FHE scheme. We also show how to build the required VFE from a standard FE together with a MAC scheme. All our results hold in the standard model. Finally, we show how one can build a verifiable computation (VC) scheme generically from a DHE. As a corollary, we get the first VC scheme which remains verifiable even if the attacker can observe verification results.

Keywords: Homomorphism Delegation, Homomorphic Encryption, Functional Encryption, Verifiable Computation.

1 Introduction

Consider a public-key encryption scenario where Alice (the sender) and Bob (the receiver) are both resource-constrained. Alice holds message m and Bob is supposed to receive $f(\mathsf{m})$ – for some function f – but neither is able to perform the associated computations. It could be the case that Alice is not even aware of which function is supposed to be applied on m, and Bob might not want to know

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 296–312, 2012.

the details of how that function is implemented. Carol is an untrusted service provider that wishes to sell computing power to Alice and/or Bob. Finally, TA is a trusted authority that is willing to take on the responsibility of approving a concrete circuit for f put forth by Carol. When function f is approved, the TA issues to Carol a token[1] TK_f that enables her to compute f on behalf of Alice and Bob, and publishes a short description h_f of the functionality provided by f to let Alice and Bob know that this is the case. When Alice encrypts a message m to Bob, the following guarantees are expected:

- If Alice and Bob are honest, then no one else will learn anything about m, as expected from a normal public-key encryption scheme (we call this property *input/output privacy*). Note that this should be the case even if the TA is honest-but-curious.
- If the TA is honest, then no one except Alice will learn anything about m, except what is leaked by f (we call this property *evaluation security*). Note that this should be the case even if Bob and Carol collude.
- Furthermore, if the TA is honest, then a successful decryption by Bob assures him that the recovered value was indeed correctly computed as $f(\mathsf{m})$ (we call this property *verifiability*).

In all cases Carol is considered to be potentially dishonest and colluding with other dishonest service providers.

For concreteness, we can picture Bob as managing the IT infrastructure in a company, and wishing to outsource e-mail filtering services to Carol. Ideally, Alice should be able to encrypt her messages to Bob, regardless of the e-mail filtering operations that will take place. Furthermore, Carol should be able to perform e-mail filtering without requiring any pre-processing assistance from Bob: Alice could even pass encrypted messages directly to Carol. Here Bob is not interested in the details of the e-mail filtering algorithm (which might not be public) and need only specify it using a short and high-level description. Furthermore, Bob trusts the TA's assurance that function f computed by Carol indeed satisfies this specification. In a natural generalisation of this scenario, multiple providers offer different e-mail filtering algorithms (or other data processing functionalities) and are all competing for Bob's preference.

In the example above, Alice could be completely oblivious of the functionality computed over her encrypted data, but this may not always be the case. In a different scenario, Bob could work for a research institute collecting health-related statistical information. Alice, working in an hospital, would be willing to collaborate with Bob and send him an encrypted database with patients' health records. However, she will only do so if she is assured that only the outputs of certain statistical measures are provided to Bob. Since neither Alice nor Bob possess the computational resources to carry out the necessary computations, they need to resort to Carol. In this case, Alice will need to rely on the TA to guarantee that tokens are only issued for functions that implement statistical

[1] Our use of the term *token* in this paper aims to distinguish secret keys associated with functions, from those associated with users.

measures that do not leak sensitive information about patients. Obviously, input/output confidentiality is an additional requirement of central importance in this setting. Furthermore, Bob's results will only be trustworthy if he can verify that Carol extracted relevant information correctly. This is also a necessary condition to assure Bob that Carol is not cheating and that she is spending the charged resources on evaluating real statistical data.

In this paper we propose and realise a new cryptographic primitive that we call Delegatable Homomorphic Encryption (DHE). The DHE primitive targets the scenarios described above, providing the required functionality and security features that permit satisfying the trust relations between Alice, Bob and Carol. This new primitive can be seen as extensions of a number of previously known cryptographic primitives: a) as a public-key counterpart to Verifiable Computation [6,2]; b) as a generalisation of Fully Homomorphic Encryption [7] (FHE) that enables control over the Evaluation operation; and c) as a delegatable (and verifiable) variant of Functional Encryption [9,11] (FE) which achieves both input and output privacy. Before presenting the DHE concept in more detail, we will explain why none of these primitives (nor trivial combinations thereof) completely solves the problems raised by the scenarios described above.

Fully homomorphic encryption. Recent breakthroughs [7] in public-key cryptography have made computing over encrypted data using homomorphic encryption a real technologic possibility. In simplistic terms, FHE allows Alice to encrypt data to Bob in such a way that Carol, knowing only the ciphertext and Bob's public key, is able to compute an arbitrary function over the encrypted plaintext without learning anything about its actual value. Confidentiality is guaranteed, and Alice does not need to be aware of the concrete functions that will be computed. Unfortunately, FHE is not intended to guarantee that Carol computes $f(\mathsf{m})$ correctly, and also does not aim to protect Alice from a collusion between Carol and Bob revealing more than $f(\mathsf{m})$.

Verifiable computation. Verifiable Computation [6] is a cryptographic primitive that takes advantage of FHE to address the important problem of computation delegation to an untrusted worker. The goal is for the Client (Bob) to delegate to the Worker (Carol) the non-interactive computation of a pre-specified function f with both confidentiality and verifiability guarantees. Bob is resource-constrained and yet should be able to: 1) prepare new inputs to the system; and 2) verify that f was correctly computed over them. This means that input preparation and output verification should involve a much lower computational cost than that required required to compute f unilaterally. Also, the computational load imposed on Carol should not be prohibitive when compared to that required to compute $f(\mathsf{m})$ in the clear. Finally, Bob should have guarantees that the data over which the computation is performed remains secret. The major difference to the DHE scenario above is that VC was designed as a symmetric primitive where Bob acts as both sender and receiver of the encrypted data. Bob can rely on a Trusted Authority to set-up the system on his behalf, but it is still assumed

that Bob must be aware of the function that is being computed by Carol when he prepares new inputs to the system.

Functional encryption. The DHE scenario also shares similarities with functional encryption [9]. This primitive is a generalisation of other cryptographic notions, including Predicate Encryption [8,12] (PE) and Attribute-Based Encryption, where ciphertexts and secret keys are associated with extra information in order to exert fine-grained control over who has access to encrypted data. In the view of FE closest to the DHE scenario [3,10], Alice hides a message m inside the ciphertext. Decryption is then accessible only to Bob if he holds a token TK_f associated with an arbitrary function f of his choice (mapping bit-strings to bit-strings) issued by a Trusted Authority. When invoked on a valid token TK_f, decryption returns $f(\mathsf{m})$. Bob should learn nothing more about the hidden message than that which is leaked by $f(\mathsf{m})$. Unfortunately, FE also comes short of solving the problems raised by the DHE scenario. The problem is that the roles of Bob and Carol are combined into a single entity, which means that Carol is essentially a trusted party if Bob is honest. Put differently, Bob is forced to either evaluate $f(m)$ through decryption, or he must delegate the decryption operation to Carol himself.

The DHE architecture. The DHE architecture is similar to the (1-hop) FHE scenario[2], extending it with the ability to control which computations can be carried out over encrypted data, and verifying their results:

- Receivers (Bob) publish public keys that can be used by senders (Alice) to prepare encrypted system inputs.
- Many senders can independently encrypt inputs to the same receiver. Encryption is independent of concrete functions that may be computed over encrypted data, and also of the evaluators (Carol) that may compute it.
- The TA is responsible for assigning computation abilities to evaluators. To enable some evaluator (Carol) to carry out computation of a specific function over encrypted data, the TA provides her with a token. This token depends only on the function that originated it, and is independent of senders and receivers.
- When running encryption, Alice computes some auxiliary information about the input. If securely shared with Bob, this permits achieving verifiability of the computed results (without leaking additional information about the original input). We discuss verifiability in more detail below.
- Finally, it is assumed that Alice and Bob are constrained in terms of computation resources, when compared to the TA and Carol. This means that encryption and decryption should be much cheaper to compute than generating tokens and evaluating functions over encrypted inputs.

[2] We consider only the case where functions computed over ciphertexts take a single input. A scenario where multiple ciphertexts from different senders can be fed to evaluation could be considered, although it is not clear what verifiability would mean in this case.

Verifiability of a decryption result must be considered in the context of a specific input and a specific function. In the DHE architecture this is captured by providing two additional inputs to decryption that set-up this context:

- The first is a short description h_f of the function f that was supposedly evaluated over the encrypted data. This is published by the Trusted Authority, and essentially binds Carol to evaluating a function f that the TA approved. In practice this description can be a hash or an identifier of a function that the TA has delegated.
- The second is a short piece of information returned by encryption and transferred securely from Alice to Bob. This auxiliary information should not reveal anything about the input data in addition to that which is revealed by the output of the computation (this is another important distinction to the VC scenario). For verifiability to be meaningful, this information must be transferred authentically. In our work we further assume that the auxiliary information is also transferred confidentially (e.g. using a combination of standard signature and encryption schemes), and leave the case where it can be known to evaluators for future work[3].

DHE construction and verifiable functional encryption. FE is a natural stepping stone towards realising our notion of a DHE. Indeed, FE already provides a means to control the functions that can be computed over encrypted data, as well as restricting the information that is leaked by decryption. However, as we described above, the features offered by FE fall short of what we require for DHE. To solve this problem, we need to consider the FE setting where decryption is no longer performed by Bob (the receiver), but is carried out on a separate module (Carol). Bob, however, must be able to efficiently check that this decryption was carried out honestly, with some help from Alice. We call this new primitive Verifiable Functional Encryption (VFE).

We construct a VFE from a standard FE scheme. The construction is intuitive – we add redundancy to function inputs and outputs to enable verification – but it is non-trivial in its realisation. Alice encrypts a MAC key k together with the message m. The tokens for a function f are now issued for a new function $f^\star$ which computes $f(\mathsf{m})$ and attaches a MAC of the result under key k. However, the security of the underlying FE scheme together with unforgeability of the MAC does *not* seem to be enough to ensure verifiability. Indeed we need special properties from the MAC scheme in order to prove that the FE scheme is preserving its authentication properties in a meaningful way. Our notion of VFE also allows for a clearer modular presentation of our results and, as we shall see, it is helpful in building VC schemes where input/output privacy is not needed.

Verifiable FE provides only a limited form of input privacy and no output privacy (with respect to Carol and the TA). To obtain a DHE scheme we therefore wrap the VFE scheme inside an FHE layer. This technique is similar to

[3] Note that although we do not consider this aspect in this paper, the hint could be anonymous in the sense that it reveals nothing about the sender. Our construction actually achieves this.

those used in [6,4] where Yao's garbled circuits and cut-and-choose protocols (instead of functional encryption) are used in conjunction with FHE. Note that although in this setting the FHE scheme must support the decryption circuit of the verifiable functional encryption scheme, it is not essential in guaranteeing verifiability. This is where the flexibility of our scheme resides.

Secure verifiable computation. A DHE scheme can be seen as a generalisation of VC where the Client functionality is now carried out by three different parties: the TA carries out the pre-computation stage, Alice prepares the inputs and Bob recovers and verifies the outputs. Not surprisingly, DHE implies a strong form of VC. In particular, the results in this paper solve two open problems identified in [6]. Firstly, any DHE scheme which is secure in the sense that we propose in this paper can be easily converted into a VC scheme that remains verifiable even if the worker (Carol) can observe the outputs of verification and adaptively choose new inputs. Secondly, if one is willing to waive input/output privacy, our construction of a VFE yields the first VC scheme that does not require FHE to achieve verifiability.

Structure of the paper. We present three security models for DHEs in Section 2. We introduce verifiable functional encryption in Section 3, and give a generic construction from a standard FE scheme and a (customised) MAC. In Section 4 we build a DHE scheme from a verifiable functional encryption scheme and a fully homomorphic encryption scheme. Relation to verifiable computation is discussed in Section 5. We conclude the paper in Section 6 by discussing the instantiation/efficiency of our construction and providing directions for future research.

2 Delegatable Homomorphic Encryption

We are now ready to formally present the syntax of a DHE scheme.

Syntax. A *delegatable homomorphic encryption (DHE) scheme* is specified by six PPT algorithms as follows.

1. $\mathsf{DHE.Setup}(1^\lambda)$: This is the setup algorithm and is run by the TA. On input a security parameter, it generates a master secret key Msk and a master public key Mpk. We assume Mpk also contains a pair of compatible message and function spaces $\mathsf{DHE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{DHE.FunSp}(\mathsf{Mpk})$.
2. $\mathsf{DHE.Gen}(\mathsf{Mpk})$: This is the key-generation algorithm and is run by the receiver. On input the master public key, it outputs a receiver key-pair $(\mathsf{sk}, \mathsf{pk})$.
3. $\mathsf{DHE.TKGen}(f, \mathsf{Msk})$: This is the token generation algorithm and is run by the TA. On input a (description of a) circuit f and the master key Msk, it generates a token TK for f and a short description of f denoted h_f. For simplicity, we restrict our attention to schemes for which h_f is deterministically generated from f and unique with overwhelming probability (e.g. using a collision-resistant hash function).

4. $\mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$: This is the encryption algorithm and is run by the sender. On input a message m, and a receiver's public key pk it returns a ciphertext c, and some auxiliary information aux.
5. $\mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})$: This is the evaluation algorithm and is run by the evaluator. On input a ciphertext c, a token TK, and a public key pk, it returns a ciphertext $\mathsf{c}_{\mathsf{evl}}$.
6. $\mathsf{DHE.Dec}(\mathsf{c}_{\mathsf{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$: This is the deterministic decryption algorithm and is run by the receiver. On input an evaluated ciphertext $\mathsf{c}_{\mathsf{evl}}$, auxiliary information aux, a short description of a function h_f, and a secret key sk, it returns either a message m or a special failure symbol $\perp$.

Recall that we assume that the TA publishes the short descriptions h_f so that they are publicly available, and that aux is supposed to be transferred securely from Alice to Bob in order to ensure verifiability.

Correctness. A DHE scheme is correct if for any $\lambda \in \mathbb{N}$, any key pair $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(\mathsf{Mpk})]$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{DHE.Gen}(\mathsf{Mpk})]$, any message $\mathsf{m} \in \mathsf{DHE.MsgSp}(1^\lambda)$ and any $f \in \mathsf{DHE.FunSp}(\mathsf{Mpk})$ we always have $f(\mathsf{m}) = \mathsf{DHE.Dec}(\mathsf{c}_{\mathsf{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$, where $(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$, $(\mathsf{c}, \mathsf{aux}) \leftarrow_{\$} \mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$, and $\mathsf{c}_{\mathsf{evl}} \leftarrow_{\$} \mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})$.

Compactness. In a compact DHE scheme the evaluation algorithm takes on the computational load of evaluating circuits over the data. This requirement, as in FHE, rules out trivial constructions where evaluation appends the circuit to ciphertexts, and it is the decryption which bears the computational load. More formally, a DHE scheme is compact if there is a fixed polynomial $\mathrm{B}(\cdot)$ such that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(1^\lambda)]$, any $f \in \mathsf{DHE.FunSp}(\mathsf{Mpk})$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{DHE.Gen}(\mathsf{Mpk})]$, any $\mathsf{m} \in \mathsf{DHE.MsgSp}(\mathsf{Mpk})$, any $(\mathsf{TK}, \mathsf{h}_f) \in [\mathsf{DHE.TKGen}(f, \mathsf{Msk})]$, any $(\mathsf{c}, \mathsf{aux}) \in [\mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})]$, and any evaluated ciphertext $\mathsf{c}_{\mathsf{evl}} \in [\mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})]$, the size of $(\mathsf{c}_{\mathsf{evl}}, \mathsf{aux}, \mathsf{h}_f, \mathsf{sk})$ is at most $\mathrm{B}(\lambda + |f(\mathsf{m})|)$ (independently of the size of f).

Outsourceability. Roughly speaking, this requirement rules out schemes which pre-compute functions at the encryption stage. Formally, a DHE scheme is outsourceable if for any $c > 0$ and sufficiently large λ, we have that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(1^\lambda)]$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{DHE.Gen}(\mathsf{Mpk})]$, and any message $\mathsf{m} \in \mathsf{DHE.MsgSp}(\mathsf{Mpk})$, and any $f \in \mathsf{DHE.FunSp}(\mathsf{Mpk})$, the time-complexity of $\mathsf{DHE.Enc}(\mathsf{m}, \mathsf{pk})$ is at most c times the time-complexity of $f(\mathsf{m})$.

Input/output privacy. We formulate a notion of input/output privacy that requires no party (including an honest-but-curious TA) except possibly the sender or the receiver should learn anything about the data enclosed in DHE ciphertexts. Formally, the TA-IND-CPA security of a DHE scheme requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ta\text{-}ind\text{-}cpa}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{TA\text{-}IND\text{-}CPA}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game TA-IND-CPA is shown in Figure 1. Note that the adversary receives Msk, which enables it to extract tokens for *all* functions. Also

note that the adversary is not given the auxiliary information aux that is used to ensure verifiability. This is consistent with the assumption in this model that the sender and the receiver are trusted.

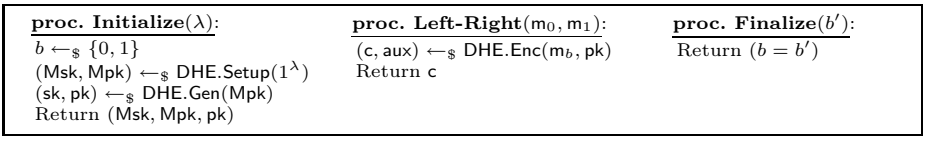


Fig. 1. Game defining the TA-IND-CPA security of a DHE scheme. An adversary is legitimate if it calls **Left-Right** exactly once on two messages of equal length.

Verifiability. Verifiability for DHEs essentially requires an evaluator to be unable to convince the sender and the receiver (sharing a secret aux that contextualises a concrete input) that it has honestly computed an incorrect value. We capture this by allowing the adversary to obtain a polynomial number of tokens for functions of its choice. We also allow the attacker to obtain a polynomial number of encrypted inputs for messages of her choice. The goal of the adversary is then to produce a ciphertext that is accepted and decrypts to an incorrect value. The model is strengthened by introducing a decryption oracle that captures the possibility that the evaluator observes the results of decryptions (verifications) executed by the receiver. More formally, VRF-CCAx security for $\mathsf{x} \in \{1, 2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{vrf\text{-}ccax}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{VRF\text{-}CCAx}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right]$$

to be negligible, where game VRF-CCAx is shown in Figure 2.

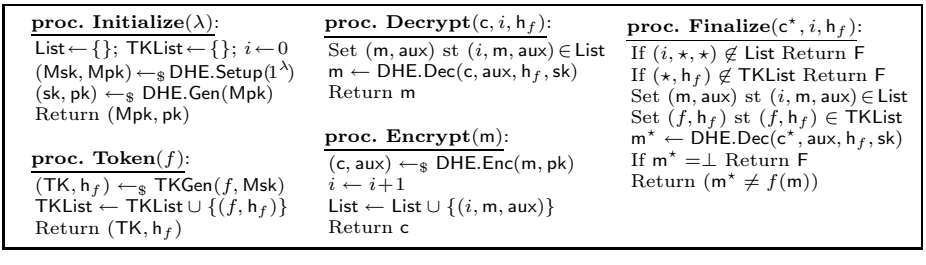


Fig. 2. Game defining the VRF-CCAx security of a DHE scheme. An adversary is legitimate if: 1) it calls **Decrypt** with an i such that $(i, \star, \star) \in \mathsf{List}$; and 2) it does not call **Token** after calling **Encrypt** when $\mathsf{x} = 1$.

A natural question that arises after introducing the previous models is how does verifiability relate to IND-CPA security. We clarify this issue in [1] by

introducing an IND-CCA security model for DHEs. In this model, where TA, Alice and Bob are assumed to be honest, the decryption oracle mimics the actions of Bob, knowing the correct secret aux for the challenge ciphertext and the set of valid function descriptions published by the TA. We show that verifiability and I/O privacy jointly imply IND-CCA security. Note that this does not mean that the DHE retains its I/O privacy in the presence of a verification oracle: the decryption oracle in the CCA security model does not allow the adversary to infer the verification result from the oracle output.

Another important feature of DHE is that it provides a reduced level of security to Alice even when Carol and Bob collude. We consider this next.

Evaluation security. Evaluation security aims to guarantee senders that no information about encrypted messages beyond that which can be obtained from evaluations and decryptions for authorised functions is leaked. This means that, even if an evaluator and a receiver collude (note that the attacker gets to see aux and the receiver's secret key) they cannot obtain more information about the original message than they should. In addition to a **Left-Right** oracle, the adversary gets a token extraction oracle to which it can only query functions that do not allow for trivial distinguishing attacks. Furthermore, the adversary may observe evaluations under other functions for all ciphertexts except the challenge. Under this notion of security, a collusion of evaluators and receivers cannot misuse their resources to compute unauthorised functions. In other words the scheme provides *resource protection*. Formally, $\mathsf{IND\text{-}EVAL}\mathsf{x}$ security for $\mathsf{x} \in \{1, 2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}eval}\mathsf{x}}_{\mathsf{DHE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}EVAL}\mathsf{x}^{\mathcal{A}}_{\mathsf{DHE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1$$

to be negligible, where game $\mathsf{IND\text{-}EVAL}\mathsf{x}$ is shown in Figure 3.

proc. Initialize(λ):
$b \leftarrow_{\$} \{0,1\}$; $\mathsf{List} \leftarrow \{\}$; $\mathsf{TKList} \leftarrow \{\}$
$(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_{\$} \mathsf{DHE.Setup}(1^{\lambda})$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{DHE.Gen}(\mathsf{Mpk})$
Return $(\mathsf{Mpk}, \mathsf{sk}, \mathsf{pk})$

proc. Token(f):
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$
$\mathsf{TKList} \leftarrow \mathsf{TKList} \cup \{f\}$
Return $(\mathsf{TK}, \mathsf{h}_f)$

proc. Evaluate(c, f):
$(\mathsf{TK}, \mathsf{h}_f) \leftarrow_{\$} \mathsf{DHE.TKGen}(f, \mathsf{Msk})$
$\mathsf{c}_{\mathsf{evl}} \leftarrow_{\$} \mathsf{DHE.Eval}(\mathsf{c}, \mathsf{TK}, \mathsf{pk})$
Return $\mathsf{c}_{\mathsf{evl}}$

proc. Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:
$(\mathsf{c}, \mathsf{aux}) \leftarrow_{\$} \mathsf{DHE.Enc}(\mathsf{m}_b, \mathsf{pk})$
$\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{c}, \mathsf{aux})\}$
Return $(\mathsf{c}, \mathsf{aux})$

proc. Finalize(b'):
Return $(b = b')$

Fig. 3. Game defining the $\mathsf{IND\text{-}EVAL}\mathsf{x}$ security of a DHE scheme. An adversary is legitimate if: 1) it calls **Left-Right** exactly one on two messages of equal length; 2) it never queries **Token** with an f such that $f(\mathsf{m}_0) \neq f(\mathsf{m}_1)$; 3) it never calls **Evaluate** with a c such that $(\mathsf{c}, \star) \in \mathsf{List}$; and 4) it does not call **Evaluate** or **Token** after calling **Left-Right** when $\mathsf{x} = 1$.

3 Verifiable Functional Encryption

As a stepping stone towards realising our DHE notion, we first introduce Verifiable Functional Encryption (VFE). Our discussion is based on the definitions of standard functional encryption [1].

Syntax. The syntax of a VFE scheme is similar to that of an FE scheme with the caveat that additional parameters are returned by the encryption, decryption and token generation algorithms that can be fed to a new verification algorithm. The verification algorithm allows anyone who is given an output of decryption to verify that it was honestly computed with the additional help of contextual information that binds the check to a specific encryption operation and a specific function that was supposedly computed over the encrypted data. Concretely, a *verifiable functional encryption (VFE) scheme* is defined through the following PPT algorithms.

1. $\mathsf{VFE.Setup}(1^\lambda)$: This is the setup algorithm. On input a security parameter 1^λ, it outputs a master public key Mpk, and a master secret key Msk. We assume Mpk also contains a pair of compatible message and function spaces $\mathsf{VFE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{VFE.FunSp}(\mathsf{Mpk})$.
2. $\mathsf{VFE.TKGen}(f, \mathsf{Msk})$: This is the token generation algorithm. On input the master secret key Msk and a function f, it outputs a token TK and a short identifier h_f for f. For simplicity, we restrict our attention to schemes for which these identifiers are deterministically generated from f and unique with overwhelming probability (e.g. via a collision-resistant hash function).
3. $\mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$: This is the verifiable encryption algorithm. On input a message m, it outputs a ciphertext c and a key k. The returned key will allow a sender to verify an output of decryption using $\mathsf{VFE.Ver}$.
4. $\mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK}, \mathsf{Mpk})$: This is the deterministic decryption algorithm. On input a token TK, a ciphertext c, and a master key Mpk, it outputs a range point/tag pair (y, t), or a special failure symbol $\perp$.
5. $\mathsf{VFE.Ver}((y, \mathsf{t}), \mathsf{h}_f, \mathsf{k}, \mathsf{Mpk})$: This is the deterministic verification algorithm. On input a range point/tag pair (y, t), a fingerprint value h_f and keys k and Mpk, it returns a decision value T or F. We also require the verification algorithm to accept only one tag for each $(y, \mathsf{h}_f, \mathsf{k})$ input.

Correctness. A VFE scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{Mpk}, \mathsf{Msk}) \in [\mathsf{VFE.Setup}(1^\lambda)]$, any $f \in \mathsf{VFE.FunSp}(\mathsf{Mpk})$, and any $\mathsf{m} \in \mathsf{VFE.MsgSp}(\mathsf{Mpk})$, we have with probability one that $f(\mathsf{m}) = y \wedge \mathsf{VFE.Ver}((y, \mathsf{t}), \mathsf{h}_f, \mathsf{k}, \mathsf{Mpk}) = \mathsf{T}$, where $(\mathsf{TK}, \mathsf{h}_f) \leftarrow_\$ \mathsf{VFE.TKGen}(f, \mathsf{Msk})$, $(\mathsf{c}, \mathsf{k}) \leftarrow_\$ \mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$, and $(y, \mathsf{t}) \leftarrow_\$ \mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK}, \mathsf{Mpk})$.

Outsourceability. A VFE scheme is outsourceable if for every $c > 0$ and sufficiently large λ we have that for any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{VFE.Setup}(1^\lambda)]$, any message $\mathsf{m} \in \mathsf{VFE.MsgSp}(\mathsf{Mpk})$, and any $f \in \mathsf{VFE.FunSp}(\mathsf{Mpk})$, the time-complexity of $\mathsf{VFE.Enc}(\mathsf{m}, \mathsf{Mpk})$ is at most c times the time-complexity of $f(\mathsf{m})$. This rules out trivial constructions which pre-compute f in encryption.

PLAINTEXT SECURITY. The IND-CCAx security of a VFE scheme for $\mathsf{x} \in \{0,1,2\}$ is defined similarly to that of a standard FE scheme, with the difference that the **Left-Right** procedure now returns (c,k). Note that this definition requires the scheme to leak nothing about the challenge except $f(\mathsf{m}_b)$ for all f for which the receiver obtains a token, even if the attacker is given k. More formally, the IND-CCAx security of a VFE scheme for $\mathsf{x} \in \{0,1,2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}ccax}}_{\mathsf{VFE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IND\text{-}CCAx}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right] - 1,$$

to be negligible, where game IND-CCAx is shown in Figure 4.

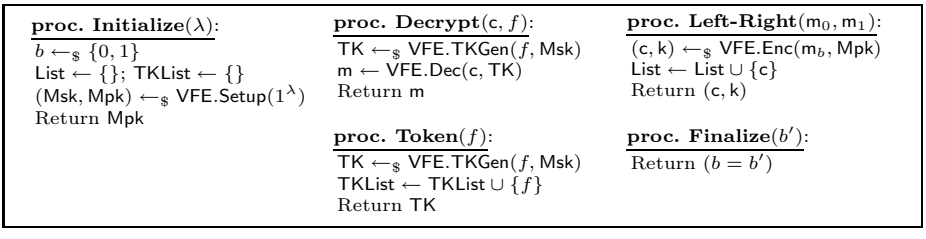


Fig. 4. Game defining the IND-CCAx security of an VFE scheme. An adversary is legitimate if: 1) it calls **Left-Right** exactly once on two messages of equal length; 2) it never calls **Decrypt** with a $\mathsf{c} \in \mathsf{List}$; 3) It never calls **Token** with an f such that $f(\mathsf{m}_0) \neq f(\mathsf{m}_1)$; 4) if $\mathsf{x} = 0$ it does not call **Decrypt**; and 5) if $\mathsf{x} = 1$ it does not call **Decrypt** after calling **Left-Right**.

VERIFIABILITY. The intuition here is similar to that in the DHE scenario: correctness of outsourced decryption should be checkable. More precisely, the VRF-CCAx security of a VFE scheme for $\mathsf{x} \in \{1,2\}$ requires the advantage of any PPT adversary $\mathcal{A}$ to be negligible, when this is defined by

$$\mathbf{Adv}^{\mathsf{vrf\text{-}ccax}}_{\mathsf{VFE},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{VRF\text{-}CCAx}^{\mathcal{A}}_{\mathsf{VFE}}(\lambda) \Rightarrow \mathsf{T}\right],$$

Here, game VRF-CCAx is shown in Figure 5. We next discuss how we construct a verifiable FE scheme.

ADDING VERIFIABILITY TO FUNCTIONS. In our construction of a VFE we use a transformation which given a pair of compatible spaces $\mathsf{FunSp}(1^\lambda)$ and $\mathsf{MsgSp}(1^\lambda)$ constructs a new function family that enables verifiability without loss of functionality – we will call this the verifiable function family. The intuition behind this construction is simple: we extend any function $f(\mathsf{m})$ to a function $f^\star(\mathsf{m},\mathsf{k})$ that takes also a secret key and computes $(f(\mathsf{m}),\mathsf{t})$ where t authenticates $f(\mathsf{m})$ under k. Formally, in order to establish a verification context that is a binding both to a concrete function and a concrete input, we take a collision resistant hash function family (CRH) and a MAC scheme (possibly

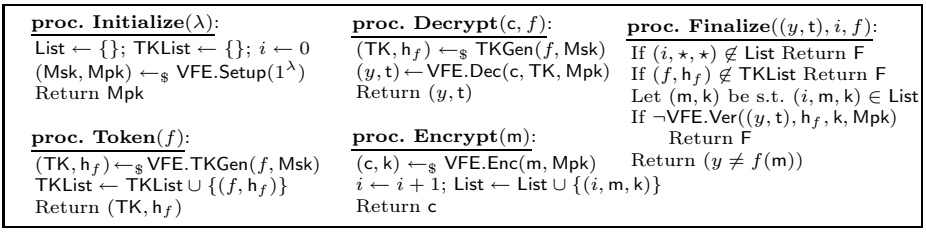


Fig. 5. Game defining the VRF-CCAx security of a VFE scheme. An adversary is legitimate it does not call **Token** or **Decrypt** after calling **Encrypt** when $\mathsf{x} = 1$.

with a global set-up procedure and a trapdoor). We use CRH to derive function fingerprints from function descriptions, and define

$$f^\star(\mathsf{m}, \mathsf{k}) := (f(\mathsf{m}), \mathsf{MAC.Tag}((f(\mathsf{m}), \mathsf{h}_f), \mathsf{k}, \mathsf{mk})),$$

where $\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$. We note that function identifiers (or fingerprints) will be unique with overwhelming probability, and that one can verify that function $f^\star$ was correctly computed by checking that $\mathsf{MAC.Tag}((y, \mathsf{h}_f), \mathsf{k}, \mathsf{mk}) = \mathsf{t}$. Also note that since MAC and CRH are deterministic the above transformation enjoys the property that $f^\star(\mathsf{m}_0, \mathsf{k}) = f^\star(\mathsf{m}_1, \mathsf{k})$ if and only if $f(\mathsf{m}_0) = f(\mathsf{m}_1)$. Furthermore given $f^\star(\mathsf{m}, \mathsf{k})$ one can read off $f(\mathsf{m})$ in time equal to $|f(\mathsf{m})|$.

The VFE construction. Our construction is simple but somewhat surprising in its capabilities. We start with any FE scheme accepting circuits for the verifiable function family that we defined above. We then show that encrypting a fresh secret key for the MAC along with the input to the function we want to compute is enough to achieve verifiability. Although this technique is intuitive, it has proven to be elusive to formalise and prove secure. Obviously, the MAC scheme must be unforgeable for one to have any hope of proving this result. However, it is not trivial to prove that including a MAC key inside a ciphertext, and issuing tokens for arbitrary functions in the above verifiable function family, does not enable the adversary to forge a MAC on the same key.

To better see where the problem arises, note that IND-CCA security of the FE scheme guarantees that no information about the input (the message and the MAC key) beyond that leaked through the extracted function(s) is leaked to an adversary. Hence, before one can reduce the verifiability of the FE to the unforgeability of the MAC one must first have a meaningful reduction to the IND-CCA security of the FE, replacing the MAC key with a different one that can be embedded in the challenge ciphertext provided to the verifiability adversary (this is because the real MAC key will not be available in the reduction to unforgeability). The trick is then to find a new key that gives rise to the same MAC, which can be substituted inside the challenge ciphertext without this being perceptible to the adversary. To achieve this result we need a MAC with the following property: it is possible to find a key which leads to identical tags on any n messages, and this key can be given to the adversary without hindering unforgeability. We call a MAC with this property an n-key-chameleon MAC.

Note that this proof-technique requires us to restrict ourselves to adversaries that perform at most n token extraction queries, all before calling the challenge. Fortunately, this is enough to ensure that our results imply a strongly secure VC scheme, as we will see later in the paper. In the next subsections we formalise this intuition, first by introducing the MACs with chameleon keys, and then describing our VFE construction in detail.

3.1 MACs with Chameleon Keys

Syntax. An *n-key-chameleon MAC scheme* is specified by three PPT algorithms as follows. The setup algorithm, $\mathsf{MAC.Setup}(1^\lambda)$, takes the security parameter and returns a pair $(\mathsf{td}, \mathsf{mk})$ consisting of a trapdoor td and public parameters mk. The deterministic tagging algorithm, $\mathsf{MAC.Tag}(\mathsf{m}, \mathsf{k})$, takes the global parameters mk, a message m, and a secret key k, and returns a tag t. Finally, the collision-finding algorithm $\mathsf{MAC.Col}(\mathsf{td}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{k})$,on input a trapdoor td, n messages, and a secret key k returns a new secret key k'.

Correctness. An n-key-chameleon MAC scheme is correct if for any $\lambda \in \mathbb{N}$, any $(\mathsf{td}, \mathsf{mk}) \in [\mathsf{MAC.Setup}(1^\lambda)]$, any $\mathsf{m}_i \in \mathsf{MAC.MsgSp}(\mathsf{mk})$ for $i = 1, \ldots, n$, any $\mathsf{k} \in \mathsf{MAC.KeySp}(\mathsf{mk})$, and any $\mathsf{k}' \in [\mathsf{MAC.Col}(\mathsf{td}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{k})]$ we have that $\mathsf{MAC.Tag}(\mathsf{m}_i, \mathsf{k}, \mathsf{mk}) = \mathsf{MAC.Tag}(\mathsf{m}_i, \mathsf{k}', \mathsf{mk})$ for all $i = 1, \ldots, n$.

Unforgeability. The $\mathsf{UF\text{-}CMA}$ security of an n-key-chameleon MAC requires the advantage of any PPT adversary $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda) \Rightarrow \mathsf{T}\right],$$

to be negligible, where game $\mathsf{UF\text{-}CMA}$ is shown in Figure 6. Note that this is essentially a slightly stronger version of a non-adaptive chosen message attack, as k' allows the adversary to compute a tag on the chosen messages.

proc. Initialize(λ):	**proc. Collision**$(\mathsf{m}_1, \ldots, \mathsf{m}_n)$:	**proc. Finalize**(m, t):
$(\mathsf{td}, \mathsf{mk}) \leftarrow_\$ \mathsf{MAC.Setup}(1^\lambda)$	$\mathsf{k}' \leftarrow \mathsf{MAC.Col}(\mathsf{td}, \mathsf{m}_1, \ldots, \mathsf{m}_n, \mathsf{k})$	If $\mathsf{m} \in \{\mathsf{m}_1, \ldots, \mathsf{m}_n\}$ Return F
$\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$	Return k'	Return
Return mk		$(\mathsf{MAC.Tag}(\mathsf{m}, \mathsf{k}, \mathsf{mk}) = \mathsf{t})$

Fig. 6. Game defining the $\mathsf{UF\text{-}CMA}$ security of an n-key chameleon MAC. An adversary is legitimate if it calls **Collision** exactly once.

Building an n-key chameleon MAC. One can build an n-key-chameleon MAC from the n-time information-theoretically secure MAC with tagging algorithm $\mathsf{MAC.Tag}(\mathsf{m}, (a_n, \ldots, a_0), \mathsf{mk}) := \sum_{i=0}^{n} a_i \mathsf{m}^i$ with $a_i, \mathsf{m} \in \mathbb{Z}_p$ for a prime p. There is no need for global parameters (except for the appropriate message and key space definitions) and no need for a trapdoor. Key generation returns a random $n+1$ tuple $(a_n, \ldots, a_0)$. To find a colliding key $\mathsf{k}' = (a'_n, \ldots, a'_0)$ given n messages, as required by the correctness condition, one solves the system of n

equations in $n+1$ unknowns. Since this system is under-defined, the new computed key can be made to look completely random. Hence, providing this key to the adversary leaks no information. A forgery as required by the game above thus translates to an n-time forgery, which is infeasible in the information-theoretical sense.

3.2 Details of the VFE Construction

Let FE be a (standard) functional encryption scheme such that for any $\lambda \in \mathbb{N}$ and any given compatible spaces $\mathsf{FunSp}(1^\lambda)$ and $\mathsf{MsgSp}(1^\lambda)$, we have that $\mathsf{FE.FunSp}(\mathsf{Mpk})$ includes $f^\star$ as defined in the equation above, for any $f \in \mathsf{FunSp}(1^\lambda)$, any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{FE.Setup}(1^\lambda)]$, any $\mathsf{hk} \in [\mathsf{CRH.KeySp}(1^\lambda)]$, and any $(\mathsf{td}, \mathsf{mk}) \in [\mathsf{MAC.Setup}(1^\lambda)]$. We define our verifiable functional encryption scheme VFE as shown in Figure 7. Note that our construction satisfies the uniqueness requirement on tags accepted by the verification algorithm by recomputing a deterministic MAC. Security properties of the scheme are stated next, and we refer to [1] for the proofs.

Theorem 1 (Informal). *The construction in Figure 7 is IND-CCA if the underlying FE scheme is IND-CCA. It is also verifiable under non-adaptive bounded attacks if the FE scheme is IND-CCA1 and the MAC is unforgeable.*

- $\mathsf{VFE.Setup}(1^\lambda)$
  1. $(\mathsf{Msk}, \mathsf{Mpk}) \leftarrow_\$ \mathsf{FE.Setup}(1^\lambda)$
  2. $(\mathsf{td}, \mathsf{mk}) \leftarrow_\$ \mathsf{MAC.Setup}(1^\lambda)$
  3. $\mathsf{hk} \leftarrow_\$ \mathsf{CRH.Gen}(1^\lambda)$
  4. Return $((\mathsf{Msk}, \mathsf{mk}, \mathsf{hk}), (\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk}))$
- $\mathsf{VFE.TKGen}(f, (\mathsf{Msk}, \mathsf{mk}, \mathsf{hk}))$
  1. $\mathsf{h}_f \leftarrow \mathsf{CRH.H}(\langle f \rangle, \mathsf{hk})$
  2. $f^\star(x_1, x_2) := (f(x_1), \mathsf{MAC.Tag}((f(x_1), \mathsf{h}_f), x_2, \mathsf{mk}))$
  3. $\mathsf{TK} \leftarrow_\$ \mathsf{FE.TKGen}(f^\star, \mathsf{Msk})$
  4. Return $(\mathsf{TK}, \mathsf{h}_f)$
- $\mathsf{VFE.Enc}(\mathsf{m}, (\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk}))$
  1. $\mathsf{k} \leftarrow_\$ \mathsf{MAC.KeySp}(\mathsf{mk})$
  2. $\mathsf{c} \leftarrow_\$ \mathsf{FE.Enc}((\mathsf{m}, \mathsf{k}), \mathsf{Mpk})$
  3. Return (c, k)
- $\mathsf{VFE.Dec}(\mathsf{c}, \mathsf{TK})$
  1. $\mathsf{m} \leftarrow \mathsf{FE.Dec}(\mathsf{c}, \mathsf{TK})$
  2. Parse $(y, \mathsf{t}) \leftarrow \mathsf{m}$
  3. Return (y, t)
- $\mathsf{VFE.Ver}((y, \mathsf{t}), \langle f \rangle, \mathsf{k}, (\mathsf{Mpk}, \mathsf{mk}, \mathsf{hk}))$
  1. Return $\mathsf{MAC.Tag}((y, \mathsf{h}_f), \mathsf{k}, \mathsf{mk}) = \mathsf{t}$

Fig. 7. A VFE scheme based on a standard FE scheme and an n-key-chameleon MAC

4 A Strongly Secure DHE Scheme

We now present a generic construction of a DHE for a given function space. The construction uses the VFE scheme introduced in the previous section as a building block. The other component in our construction is an FHE scheme. For the definitions and security models of FHE we refer the reader to [1].

Let FHE be a compact homomorphic public-key encryption scheme (supporting arity-1 functions), and let VFE be an outsourceable verifiable functional encryption scheme for the function family constructed in the previous section. Suppose also that for any $\lambda \in \mathbb{N}$, any $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{FHE.Gen}(1^\lambda)]$ and any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{VFE.Gen}(1^\lambda)]$, the following compatibility conditions hold:

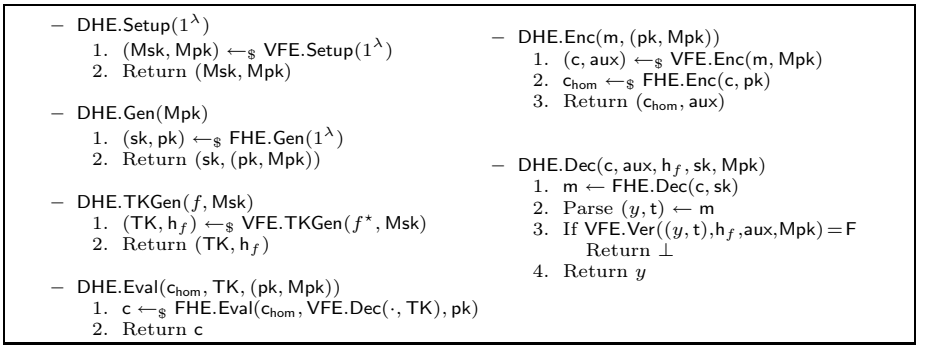


Fig. 8. A DHE scheme based on an FHE and a VFE scheme

$\mathsf{FHE.MsgSp}(\mathsf{pk}) = \mathsf{VFE.CphSp}(\mathsf{Mpk})$ and $\mathsf{VFE.Dec}(\cdot,\cdot) \in \mathsf{FHE.FunSp}(\mathsf{pk})$. We construct an outsourceable DHE scheme as shown in Figure 8. We define $\mathsf{DHE.MsgSp}(\mathsf{Mpk})$ and $\mathsf{DHE.FunSp}(\mathsf{Mpk})$ as arbitrary spaces that may parameterise our function family construction.

The correctness of the above scheme follows from the correctness of the underlying FHE and VFE schemes. The compactness of the construction follows from the compactness of the underlying FHE scheme, whereas its outsourceability follows from the outsourceability of the VFE. The security guarantees of the scheme are stated next. The precise statement and proof of the theorem is presented in [1].

Theorem 2 (Informal). *The DHE construction in Figure 8 provides input/output privacy, verifiability, and evaluation security if the VFE scheme is IND-CCA and verifiable, and the FHE is IND-CPA.*

5 Secure Verifiable Computation from DHE

Converting a DHE scheme to a VC scheme is straightforward. Suppose DHE is a delegatable homomorphic encryption scheme. For any $\lambda \in \mathbb{N}$, and any $(\mathsf{Msk}, \mathsf{Mpk}) \in [\mathsf{DHE.Setup}(1^\lambda)]$ set $\mathsf{VC.FunSp}(1^\lambda) := \mathsf{DHE.FunSp}(\mathsf{Mpk})$ and $\mathsf{VC.FunSp}(1^\lambda) := \mathsf{DHE.MsgSp}(\mathsf{Mpk})$ (we assume the spaces returned by DHE depend only on λ). Our construction is given in Figure 9. The correctness of the above scheme follows from the correctness of the underlying DHE scheme. Outsourceability of the VC scheme follows form the compactness and outsourceability of the DHE scheme. The security properties of this scheme are given next. The precise statements and proofs may be found in [1].

Theorem 3 (Informal). *The VC scheme in Figure 9 is input/output private and* fully *verifiable if the underlying DHE scheme is input/output private and non-adaptively verifiable.*

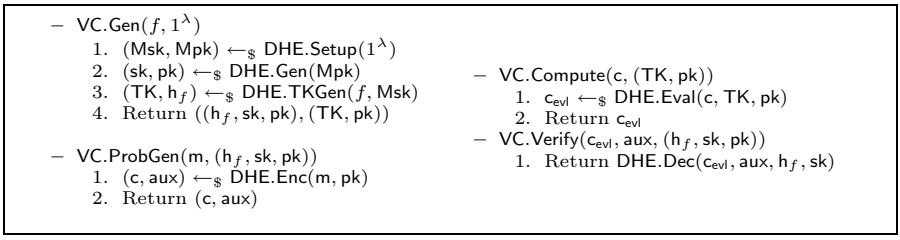


Fig. 9. A VC scheme based on a DHE scheme

It follows directly from this result that our DHE constructions implies the first VC scheme that preserves verifiability even if the adversary can adaptively observe verification results. Furthermore, our scheme preserves verifiability even if one removes the FHE layer. Hence, our results also answer an open question in [6]: it is possible to obtain a VC scheme without the cost of an FHE scheme, if one is willing to sacrifice I/O security.

Remark. As for DHEs, we note that the VC scheme above does not provide input/output privacy in the presence of verification oracle. In scenarios where this level of confidentiality is required, a DHE scheme which provides CPA security in the presence of a verification oracle is needed. We leave this as a direction for future work.

6 Concluding Remarks

Instantiating our DHE construction. Currently, no fully secure PE scheme for general predicates exists, and hence our construction can still not be realised in its *full* generality. The recent scheme of De Caro et al. [5], however, does allow for the instantiation of our construction for a restricted class of functions. We refer the interested reader to [1] for the details.

Open problems. An interesting direction is to uncover solutions that go beyond the non-adaptive and/or bounded token-extraction level of security that our DHE scheme achieves. Simulation-based notions of security, given the impossibility result of Boneh et al. [3], deserve further investigation. In terms of functionality, the proposed DHE primitive is somewhat limiting in what it offers: it is intrinsically 1-hop, and works only for functions of arity one; its delegation capabilities are not hierarchical; and its encryption routine is public and cannot offer function privacy for tokens. We leave such extensions for future work.

Acknowledgments. We would like to thank anonymous reviewers, and particularly an ASIACRYPT 2011 reviewer, for their valuable comments. Manuel Barbosa was supported by project SMART, funded by ENIAC Joint Undertaking (GA 120224). Pooya Farshim was supported in part by grant Fi 940/4-1 of the German Research Foundation (DFG). While at Royal Holloway, Univeristy of London, Pooya Farshim was also sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001.

References

1. Barbosa, M., Farshim, P.: Delegatable Homomorphic Encryption with Applications to Secure Outsourcing of Computation. Cryptology ePrint Archive, Report 2011/215
2. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable Delegation of Computation over Large Datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
3. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
4. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
5. De Caro, A., Iovino, V., Persiano, G.: Efficient Fully Secure (Hierarchical) Predicate Encryption for Conjunctions, Disjunctions and k-CNF/DNF Formulae. Cryptology ePrint Archive, Report 2010/492 (2010)
6. Gennaro, R., Gentry, C., Parno, B.: Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
7. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: 41st ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178 (2009)
8. Katz, J., Sahai, A., Waters, B.: redicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
9. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
10. O'Neill, A.: Definitional Issues in Functional Encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
11. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
12. Shen, E., Shi, E., Waters, B.: Predicate Privacy in Encryption Systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)

Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting*

Carmit Hazay[1], Gert Læssøe Mikkelsen[2,**], Tal Rabin[3], and Tomas Toft[1]

[1] Department of Computer Science, Aarhus University, Denmark
{carmit,ttoft}@cs.au.dk
[2] The Alexandra Institute
gert.l.mikkelsen@alexandra.dk
[3] IBM T.J.Watson Research Center
talr@us.ibm.com

Abstract. The problem of generating an RSA composite in a distributed manner without leaking its factorization is particularly challenging and useful in many cryptographic protocols. Our first contribution is the first non-generic *fully simulatable* protocol for distributively generating an RSA composite with security against malicious behavior in the two party setting. Our second contribution is a *complete* Paillier [37] threshold encryption scheme in the two-party setting with security against malicious behavior. Our RSA key generation is comprised of the following: (**i**) a distributed protocol for generation of an RSA composite, and (**ii**) a biprimality test for verifying the validity of the generated composite. Our Paillier threshold encryption scheme uses the RSA composite as public key and is comprised of: (**i**) a distributed generation of the corresponding secret-key shares and, (**ii**) a distributed decryption protocol for decrypting according to Paillier.

1 Introduction

Generation of RSA Composite. Generating an RSA composite, N, (a product of two primes, p and q), and secret keying material (values related to $\phi(N)$) in a distributed manner is an important question in secure computation. Many cryptographic protocols require such a composite for which none of the parties know its factorization. A concrete example where such a protocol is very useful is *threshold cryptography* where a number of parties exceeding a threshold is required to cooperate in order to carry out a cryptographic task; see [39,21,41,13] for just a few particular examples. Another important application is using this composite for securely evaluating any function in the common reference string model (CRS) in a generic form [34], or functions of specific interests such as the Fiat-Shamir authentication protocol [24,23], set-intersection [33] and oblivious pseudorandom functions [33].

This computation has proven particularly challenging and most prior works assumed that the composite is generated by a *trusted dealer*. In a breakthrough

* The full version of this paper can be found in [31].
** This work was partially done at the Dept. of Computer Science, Aarhus University.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 313–331, 2012.

result, Boneh and Franklin [4] showed a mathematical method for choosing a composite and verifying that it is of the proper form. Based on this method they designed a secure protocol in the multiparty setting for an honest-but-curious adversary with honest majority. Frankel et al. [26] strengthened this result to be resistant to a fully malicious adversary. Additional solutions for testing primality in the multiparty setting appear in [1,18].

The two party setting posed additional barriers even in the semi-honest model. Cocks [10] initiated the study of the shared generation of the RSA composite in the two-party semi-honest model. But the proposed protocol was later found to be insecure [11,3]. The problem was finally solved by Gilboa [30] who presented a protocol in the semi-honest model. In the malicious setting, Blackburn et al. [3] described an active secure protocol, however, they do not provide a proof of security for their protocol. Concurrently, Poupard and Stern [38] proposed a solution that runs in time proportional to the size of the domain from which the primes are sampled, which is exponential in the security parameter. Furthermore, a second minor limitation is a requirement that one step has to be computed in complete simultaneity. This can probably be fixed by using commitments. They attempt to somewhat reduce the run time by introducing various modifications, however, those are not proven, and as they leak some information presenting a proof of security (if at all possible) will not be trivial. A detailed discussion appear in the full version of this paper [31]. Thus, all these results do not offer an efficient and provable solution for the two-party malicious case.

Our First Result. The first RSA key generation in the malicious setting which is an efficient (non-generic) protocol with a full simulation based proof of security.

We define the appropriate functionalities and prove that our protocols realize them. Our formalization takes into account a subtle issue in the RSA key generation, which was noticed by Boneh and Franklin [4]. Informally, they showed that their protocol leaks certain amount of information about the product and further proved that it does not pose any practical threat for the security. Nevertheless, it does pose a problem when simulating, we therefore choose to work with a slightly modified version of the natural definition for a key generation. Our scheme is comprised of the following protocols:

1. **A Distributed Generation of an RSA Composite.** We present the first fully simulatable protocol for securely computing a composite as a product of two primes without leaking information about its factorization. Our protocol follows the outlines of [4] and improves the construction suggested by [30] in terms of security and efficiency. We introduce additional trial division which significantly improves efficiency. [30] did not achieve trial division in the two party setting, thus our solution is the first in the two-party setting.
2. **A Distributed Biprimality Test.** We adopt the biprimality test proposed by [4] into the malicious two-party setting. This test essentially verifies whether the generated composite is of the correct form. We provide a proof of security for this protocol.

Our Second Result. The first threshold Paillier [37] encryption scheme in the malicious setting which is an efficient, (non-generic) protocol with a full simulation based proof of security.

Threshold Paillier. A threshold cryptosystem usually involves two related yet separable components; (**1**) a distributed generation of the public keys and a sharing of the secret key and, (**2**) a decryption/signature computation from a shared representation of the secret key. Solutions for distributed discrete log based system key generation [28], threshold encryption/decryption for RSA [29,41], DSS [27] and Paillier [25,15,2] in the multiparty setting have been presented. For some schemes the techniques from the multiparty setting can be adapted to the two-party case in a more-or-less straightforward manner. However, the case of malicious two-party Paillier has proven more complex and elusive.

More specifically, as the RSA and Paillier encryption schemes share the same public key/secret key format of a composite N and its factorization, it may seem at first glance that decryption according to Paillier should follow the outlines as decryption according to RSA as e.g., in [8]. We observe that when decrypting as in RSA (i.e., raising the ciphertext to the inverse of N modulo the unknown order), the decrypter must extract first the randomness used for computing the ciphertext in order to complete the decryption. This property may be problematic in the context of simulation based secure computation because it forces the simulator to present the randomness of the ciphertext instead of proving correctness using ZK proofs. In addition we recall that by definition Paillier's scheme requires extra computation in order to complete the decryption (on top of raising the ciphertext to the power of the secret value) since the outcome from this computation is the plaintext multiplied with this secret value. In the distributive setting this implies that the parties must keep two types of shares. Our threshold scheme is comprised of the following protocols:

1. **Distributed Generations of the Secret Key Shares.** We present protocols for generating additive and multiplicative shares for $\phi(N)$. The generation of the initial additive shares follows from the key generation almost immediately, whereas generating the multiplicative shares is more challenging.
2. **A Distributed Decryption.** Motivated by the discussion above, we present a distributed protocol for decrypting according to Paillier. Our protocol takes a different approach than traditionally proving computations using zero-knowledge proofs.

Efficiency. We provide a detailed efficiency analysis for our subprotocols in Section 5. Roughly speaking, all subprotocols have constant round complexity due to parallelization of the generation of RSA composites, including the biprimality tests and trial division, and the decryption of multiple instances. Moreover, all our zero-knowledge proofs (except one that achieves constant complexity on the average) run in constant rounds and require constant number of exponentiations.

In the full version [31] we further show how to extend our protocols to the multiparty setting with dishonest majority, presenting the first actively secure k-party RSA generation protocol, tolerating up to $k-1$ corruptions.

2 Preliminaries

We denote the security parameter by 1^n. We focus our attention on the following, widely used, variant of Paillier comprised of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm, Gen, chooses two equal length primes p and q and outputs a public key $pk = N = pq$, and a matching secret-key $sk = \phi(N)$. The encryption procedure of a message $m \in \mathbb{Z}_N$, denoted $\mathsf{Enc}_{pk}(m)$, is performed by choosing $r \in_R \mathbb{Z}_N^*$ ($\mathbb{Z}_N$ in practice), and computing $(1+N)^m \cdot r^N \bmod N^2$, whereas decrypting, $\mathsf{Dec}_{sk}(c) = \frac{[c^{\phi(N)} \bmod N^2]-1}{N} \cdot \phi(N)^{-1} \bmod N$. The security of Paillier is implied by the Decisional Composite Residuosity Assumption (DCR). Two additional building blocks used are: integer commitment schemes, an example is the Paillier based commitment scheme of Damgård and Nielsen [19,20]; and ElGamal encryption [22] secure under the DDH assumption. An additive homomorphic version of ElGamal encryption is used, where g^m and not m is encrypted, g being the generator of the group. Both a distributed key generation protocol (π_{GEN}), and a distributed decryption protocol (π_{DEC}) exists, see [32].

3 A Distributed Generation of an RSA Composite

This section presents the protocol, DKeyGen for distributively generating an RSA composite without disclosing information about its factorization and with security against malicious activities. In DKeyGen the parties generate candidates for the potential composite which they run through a biprimality test for checking its validity. Our protocol is useful for designing distributive variants of the RSA encryption and signature schemes, as well as other schemes that rely on factoring related hardness assumptions. In this paper the protocol is used for generating keys for our threshold Paillier encryption scheme. The starting point for DKeyGen is the protocols of [4,30]. These protocols are designed to distributively generate an RSA composite $N = pq$ with an unknown factorization. The protocol by Boneh and Franklin [4] assumes honest majority, whereas the protocol by Gilboa [30] adopts ideas from [4] into the two-party setting; both are secure in the semi-honest setting.

Recall that when coping with malicious adversaries it must be assured that the parties follow the protocol specifications. In our context this means that the parties' shares must be of the appropriate length and that no party gains any information about the factorization of N. This challenging task is typically addressed by adding commitments and zero-knowledge proofs to each step of the protocol. Unfortunately, this is usually not very practical since the statements

that needed to be proven are complicated, and therefore leading to highly inefficient protocols. Instead, we will be exploiting specific protocols for our tasks (some new to this work), that are both efficient and fully secure. By proper analysis of where to use zero knowledge proofs, which proofs to use and moreover, by a novel technique of utilizing two different encryption schemes with different homomorphic properties, we achieve a highly efficient key generation protocol. It should be noted that except for the setup step which is only executed once we can avoid expensive zero knowledge proofs based on the cut and choose technique. Additional optimizations can be found in Section 5.

For the sake of completeness we include a short description of [4] as adapted by [30] for the two-party setting. These protocols consist of the following three steps: **1**) Each party P_i generates two random numbers p_i and q_i representing shares of p and q, such that $p = \sum_i p_i$, $q = \sum_i q_i$ and $p \equiv q \equiv 3 \pmod 4$. We note that [4] includes a distributed trial division of p and q for primes less than a bound B, which greatly improves the efficiency of the protocol. Trial division is not obtained by [30], making our solution the first two party protocol achieving the significant speedup from trial division. **2**) After having created the two candidates for being primes the parties execute a secure multiplication protocol to compute $N = (p_0 + p_1)(q_0 + q_1)$. In [4] this step is based on standard generic solutions. Here we take a novel approach of utilizing both ElGamal and Paillier encryption schemes, giving us active security at a very low cost. Generating the RSA composite this way does not guarantee that the composite is made of uniformly random primes since the adversary can, in some limited way, influence the distribution of the primes. This issue was observed in [4] and discussed further below. **3**) Finally, the candidate N for being an RSA composite is tested by a distributed biprimality test, which requires $p \equiv q \equiv 3 \pmod 4$. If the test rejects N as being a proper RSA modulus, the protocol is restarted.

Typically, the definition for the key generation algorithm requires that the RSA composite is a product of two randomly chosen equal length primes p and q. However, due to the distributed biprimality test N must be a Blum integer ($N = pq$, where $p \equiv q \equiv 3 \mod 4$). This is a common requirement for distributed (bi)primality tests and does not decrease the security, since about $1/4$ of all RSA modulus are Blum integers.

As observed by [4], the parties can slightly influence the distribution of each prime, because p and q are generated by adding shares over the integers which implies that they are not uniformly random. So each party has some (limited) knowledge of the distribution based on its shares. We define the public key generation algorithm, Gen', capturing this deviation and generating N by the same distribution as the protocol. This is obtained by Gen' receiving additional two inputs r_p and r_q, representing a potential adversary's input shares. Gen' adds r_p and r_q with randomly chosen shares and ensures that the sum is congruent to $3 \mod 4$. Formally, let $\mathsf{Gen}'(1^n, r_p, r_q)$ denote a public key generation that takes two additional inputs besides the security parameter 1^n and works as follows:

1. If $r_p, r_q \geq 2^{n-2}$ output $\perp$ and halt.
2. Otherwise, choose a uniform random $s_p \in \{0,1\}^{n-2}$.
3. Calculate $p = 4(r_p + s_p) + 3$ and examine the outcome:
   - if p is composite, then goto Step 2 and choose a new value for s_p.
   - if p is prime, then repeat the process to generate q.
4. Return $N = pq$, and generate the private key as in Gen.

As proved by Boneh and Franklin, using Gen' instead of Gen does not give the adversary the ability to factor N even if it can slightly influence its distribution.

Functionality $\mathcal{F}_{\text{GEN}}$

Key Generation: Upon receiving from party P_i message $(\mathsf{Generate}, 1^n)$, $\mathcal{F}_{\text{GEN}}$ sends $(\mathsf{RandInput})$ to the adversary and waits for the the reply $(\mathsf{GenInput}, r_a, r_b)$ from the adversary . $\mathcal{F}_{\text{GEN}}$ then invokes $(pk, sk) \leftarrow \mathsf{Gen}'(1^n, r_a, r_b)$, records sk and sends pk to the adversary. If the adversary replies allow, then $\mathcal{F}_{\text{GEN}}$ sends pk to the parties, ignoring further messages of this form. Otherwise, it sends $\perp$ to the honest party.

Fig. 1. The RSA Modulus Generation Functionality

Protocol $\mathsf{DKeyGen}$ includes in addition to the previously mentioned three steps a *key-setup* step used to generate keys for commitment and encryption schemes. A shared key is generated for the distributed additively homomorphic ElGamal encryption scheme and each party generates a key for standard non-distributive Paillier encryption and integer commitments. We are using both ElGamal and Paillier due to efficiency considerations, because most zero-knowledge proofs used here can be implemented in an efficient manner when applied on ElGamal (with a known group order), rather than on Paillier. Nevertheless, the plaintext cannot be recovered efficiently and therefore we use Paillier in a non-distributive fashion. In addition, the fact that a distributive ElGamal variant can be easily obtained allows us to design a trial division test that is run on individual primes and improves the numbers of trials. In order to cope with malicious adversaries, our protocols employ zero-knowledge (ZK) proofs. Some of these are known, others are new to this work and are interesting by themselves. We note that all the proofs that participate in Protocol 1 require a strict constant overhead. In Appendix A we specify these ZK proofs in detail.

Protocol 1. [$\mathsf{DKeyGen}$] *Distributed generation of RSA composite with active security:*

- **Inputs for parties** P_0, P_1**:** 1^n *and a threshold* B *for the trial division.*

1. **Key-Setup**
   (a) *The parties run protocol* π_{GEN} *for generating a public key* $pk_{\text{EG}} = (g, h)$ *and secret key shares* sk^0_{EG} *and* sk^1_{EG} *for ElGamal.*

(b) *Each party P_i generates a Paillier key pair $(pk^i_{\text{Pa}}, sk^i_{\text{Pa}})$ with a modulus bit length $\lambda > 2n$, and sends $pk^i_{\text{Pa}} = N^i_{\text{Pa}}$ to the other party. Each party proves correctness of N^i_{Pa} by π_{RSA} (cf. Appendix A). The Paillier keys are used for encryptions and commitments.*

2. **Generate Candidates**

(a) **Generate Shares of Candidate.** *Each party P_i picks a random $(n-2)$-bit value $\bar{p}_i$, encrypts it and sends $\bar{c}_i = \mathsf{Enc}_{pk_{\text{EG}}}(\bar{p}_i)$ to the other party. The parties prove knowledge of the plaintexts, via π_{ENC}, and prove that $\bar{p}_i < 2^{n-2}$ via π_{BOUND}.*
In order to ensure that $p_0 \equiv 3 \bmod 4$, the parties compute $c_0 \leftarrow (\bar{c}_0)^4 \cdot \mathsf{Enc}_{pk_{\text{EG}}}(3)$. Similarly, the parties ensure that $p_1 \equiv 0 \bmod 4$ by $c_1 \leftarrow (\bar{c}_1)^4$.

(b) **Trial division.** *For all primes $\alpha \leq B$, the parties run trial division on $p = p_0 + p_1$. Each party P_i sends an encryption $c_i^{(\alpha)} = \mathsf{Enc}_{pk_{\text{EG}}}(p_i \bmod \alpha)$ to the other party, and proves the correctness of the computation using π_{MOD}.*
The parties compute $c^{(\alpha)} \leftarrow c_0^{(\alpha)} \cdot c_1^{(\alpha)}$ and $\tilde{c}^{(\alpha)} \leftarrow c^{(\alpha)} \cdot \mathsf{Enc}_{pk_{\text{EG}}}(-\alpha)$. Clearly α divides p if and only if $p_0^{(\alpha)} + p_1^{(\alpha)} \in \{0, \alpha\}$, i.e. when either $c^{(\alpha)}$ or $\tilde{c}^{(\alpha)}$ is an encryption of zero. This is checked by raising these to secret, non-zero exponents and decrypting. If no prime $\alpha < B$ divides the candidate it is accepted by trial division.

(c) **Repeat.** *Repeat Steps 2a- 2b until two candidates p and q survive trial division.*

3. **Compute Product ($N = pq$)**

(a) **Compute the product.** *P_0 sends P_1 encryptions of $\tilde{p}_0 = p_0$ and $\tilde{q}_0 = q_0$ under pk^0_{Pa} and proves knowledge of plaintexts using π_{ENC}. (Note that a malicious P_0 may send encryptions of incorrect values). Next, P_1 computes and sends:*

$$\begin{aligned} c_{\tilde{N}-\tilde{p}_0\tilde{q}_0} &\leftarrow \mathsf{Enc}_{pk^0_{\text{Pa}}}(p_0)^{q_1} \cdot \mathsf{Enc}_{pk^0_{\text{Pa}}}(q_0)^{p_1} \cdot \mathsf{Enc}_{pk^0_{\text{Pa}}}(p_1 q_1) \\ &= \mathsf{Enc}_{pk^0_{\text{Pa}}}((p_0+p_1)(q_0+q_1) - p_0 q_0) \end{aligned}$$

Furthermore, P_1 proves that $c_{\tilde{N}-\tilde{p}_0\tilde{q}_0}$ has been computed as a known linear combination based on $\mathsf{Enc}_{pk^0_{\text{Pa}}}(\tilde{p}_0)$ and $\mathsf{Enc}_{pk^0_{\text{Pa}}}(\tilde{q}_0)$ using π_{VERLIN}. P_0 decrypts, thus obtaining the plaintext $m_{\tilde{N}-\tilde{p}_0\tilde{q}_0}$; from this $\tilde{N} = m_{\tilde{N}-\tilde{p}_0\tilde{q}_0} + \tilde{p}_0\tilde{q}_0$ is computed and sent to P_1 along with an encryption $c_\pi = \mathsf{Enc}_{pk^0_{\text{Pa}}}(\tilde{p}_0\tilde{q}_0)$. Finally, using π_{MULT} and π_{ZERO}, P_0 proves that c_π contains the product of the two original ciphertexts and that $\tilde{N}$ is the plaintext of: $c_{\tilde{N}-\tilde{p}_0\tilde{q}_0} c_\pi = \mathsf{Enc}_{pk^0_{\text{Pa}}}((\tilde{p}_0 + p_1)(\tilde{q}_0 + q_1))$.

(b) **Verify Multiplication.** *The parties use the homomorphic property of ElGamal encryption to compute an encryption of $N = (p_0+p_1)(q_0+q_1)$ from the ciphertexts generated at Step 2a. The computation is analogous to that of Step 3a, again using π_{MULT} for proving correct multiplication of $(p_i \cdot q_i)$*
The parties use secure decryption of distributed ElGamal π_{DEC} to obtain g^N, and both verify that $g^{\tilde{N}} = g^N$, i.e. that $N = \tilde{N}$ and abort if $g^{\tilde{N}} \neq g^N$.

4. **Biprimality Test**
Execute biprimality test (cf. Section 3.1) and accept N if the test has accepted, otherwise the protocol is restarted from Step 2a.

Theorem 1. *Assuming hardness of the DDH and DCR problems, Protocol 1 realizes $\mathcal{F}_{\text{GEN}}$ in the presence of malicious adversaries.*

Proof Overview. If both parties follow the protocol then a valid RSA modulus N is generated with high probability. In the last iteration of the protocol two elements are chosen randomly and independently of previous generated candidates and are multiplied to produce N. By the correctness of the biprimality test specified below, N is a product of two primes with overwhelming probability.

We assume the simulator $\mathcal{S}$ has knowledge of the distribution of the loops in the protocol. $\mathcal{S}$ can simulate this by running the protocol "in its head", emulating the role of the honest party. Namely, denoting by P_i the corrupted party, then upon extracting the adversary $\mathcal{A}$'s shares p_i, q_i, $\mathcal{S}$ picks two shares p_{1-i}, q_{1-i} as the honest party would do and checks whether $N_{\mathcal{S}} = (p_i + p_{1-i})(q_i + q_{1-i})$ is a valid RSA composite. If not, then this is not the final iteration of the protocol, and $\mathcal{S}$ uses p_{1-i}, q_{1-i} to perfectly emulate the role of the honest P_{1-i}. If this is the final iteration, $\mathcal{S}$ asks the trusted party for $\mathcal{F}_{\text{GEN}}$ to generate an RSA composite with p_i, q_i being $\mathcal{A}$'s input and completes the execution by emulating the role of the honest party on arbitrary shares. The simulation is different for the two corruption cases as the parties' role is not symmetric. If P_0 is corrupted, $\mathcal{S}$ sends back in Step 3a the encryption of the composite returned from the trusted party and makes the ElGamal decryption decrypt into this composite as well. If P_1 is corrupted $\mathcal{S}$ "decrypts" the Paillier ciphertext result into that composite and then makes the ElGamal decryption return the same outcome. In Step 3a, where the parties compute the product, it is insufficient to let P_1 complete the computation over the encrypted shares of P_0 without verification of correctness. The problem is that P_1 may attempt to compute N in a different, potentially failing way. Hence if it *finds* N, this may leak information. This issue might not seem critical for practical considerations, however, it is for simulation based security. This makes this corruption case a particular challenge, however, the full proof show how an alternative computation in a *successful* execution implies guessing the shares of the honest party before the RSA-modulus is revealed. The complete proof, in the full version of this paper [31] is done by a series of games.

3.1 The Biprimality Test

The distributed biprimality test is based on a test by Boneh-Franklin [4] where the parties agree on a random element $\gamma \in \mathbb{Z}_N^*$ with Jacobi symbol 1, and raise γ to a power calculated from their shares. The test accepts a number with more than two prime factors with probability at most 1/2. Therefore, the parties repeat the test sufficiently many times in order to decrease the error. We adopt this test for the malicious setting. The biprimality test by Damgård and Mikkelsen [18] has a better error estimate, however, it cannot be used directly in the two-party setting with malicious adversaries. In the full version of this paper we adapt their test into the two-party setting with semi-honest adversaries.

Protocol 2. [DPrim] *A distributed biprimality test:*

- **Inputs:** 1^n, *a statistical parameter* 1^ℓ *and a public key candidate* N.
- **The Protocol:**
  1. *The parties jointly generate a random element* $\gamma \in \mathbb{Z}_N^*$ *with Jacobi symbol* $\mathcal{J}(\gamma) = 1$. *By standard techniques this is made secure against active deviation.*

2. *The parties compute the encryption* $e_0 = \mathsf{Enc}_{pk_{\mathrm{EG}}}\left(\frac{N-(p_0+q_0)+1}{4}\right)$ *using the homomorphic property of ElGamal* (P_1 *knows the encryptions of* p_0 *and* q_0 *from the earlier protocol). Furthermore,* P_0 *sends* $\gamma_0 = \gamma^{\left(\frac{N+1-(p_0+q_0)}{4}\right)} \bmod N$ *and proves consistency between* e_0 *and* γ_0 *using* π_{EQ}.
3. P_1 *sends* $\gamma_1 = \gamma^{\left(\frac{-(p_1+q_1)}{4}\right)} \bmod N$ *to* P_0 *and proves consistency using* π_{EQ} *to an encryption* e_1 *of* $\frac{-(p_1+q_1)}{4}$*, computed as above.*
4. *Finally, the parties reject* N *if and only if* $\gamma_0 \cdot \gamma_1 \bmod N \neq \pm 1$*. We further note that the test by [4] includes an additional step were instead of using* γ*, the parties randomly pick an element from the group* $(\mathbb{Z}_N[x]/(x^2+1))^*/\mathbb{Z}_N^*$*; we omit the details due to the similarity of the above test.*
5. *This test is repeated* ℓ *times to achieve sufficiently small error.*

Lemma 1. *Assuming hardness of the DDH problem, Protocol 2 is a distributed* Monte Carlo *algorithm such that on a statistical parameter* 1^ℓ *and a random* γ*, it holds that: 1) Correctly formed RSA moduli* $N = pq$*, where* $p \equiv q \equiv 3 \pmod 4$ *are always accepted. 2) The average case probability of accept if either* p *or* q *is a composite, is at most* $2^{-\ell}$*. 3) The protocol is simulatable without knowledge of the factorization of* N *in the presence of malicious adversaries.*

Correctness follows from [4]. Security follows as the simulator $\mathcal{S}$ is able to simulate by having knowledge of the adversary's shares and thereby being able to calculate γ_0 or γ_1. More specifically, observe that there are two possible outcomes of DPrim. N being rejected is simulated by $\mathcal{S}$ choosing shares on behalf of the honest player such that N is not a Blum integer and executing the real protocol. In the case where N should be accepted it is a Blum integer. This implies enough knowledge of $\mathbb{Z}_N^*$ for $\mathcal{S}$ to chose the simulated γ as a uniform random element in $\mathbb{Z}_N^*$ with $\mathcal{J}(\gamma) = 1$. A complete proof is found in the full version [31].

4 A Complete Threshold Paillier Cryptosystem

In the following section, we describe our threshold construction in the two-party setting for the Paillier encryption scheme [37]. Our Threshold Paillier Scheme, **TPS**, is comprised of the following subprotocols: (**i**) $\mathsf{DKeyGen}$ for distributed generation of an RSA composite. (**ii**) Dsk for distributed generation of the corresponding secret-key shares. (**iii**) DDec for distributed decryption according to Paillier's specifications while maintaining the randomness of the ciphertext a secret. These protocols rely on the following standard hardness assumptions: (**1**) DDH due to employing the ElGamal scheme [22] and composite DDH due to [17], and (**2**) DCR due to employing the Paillier scheme [37] and integer commitments [19,20].

Our protocols form the *first complete threshold scheme for Paillier* in the two-party setting with security in the presence of malicious adversaries under full simulation based definitions, following the ideal/real model paradigm. We denote by $\Pi = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ the Paillier encryption scheme, with the modified key generation algorithm Gen' specified in Section 3, encryption algorithm Enc and decryption algorithm Dec. The formal description of the threshold functionality, $\mathcal{F}_{\mathrm{THRES}}$ is found in Figure 2.

Theorem 2. *Assuming hardness of the (composite) DDH and DCR problems, the scheme* $\mathbf{TPS} = (\mathsf{DKeyGen}, \mathsf{Dsk}, \mathsf{DDec})$ *computes functionality* $\mathcal{F}_{\text{THRES}}$ *in the presence of malicious adversaries.*

The proof follows from the proofs for Lemma 1 and Theorems 1, 3 and 4.

Functionality $\mathcal{F}_{\text{THRES}}$

KEY GENERATION: Identical to KEY GENERATION in $\mathcal{F}_{\text{GEN}}$.
DECRYPTION: Upon receiving a $(\mathsf{Decrypt}, c, \mathsf{Both})$ message from party P_i, $\mathcal{F}_{\text{THRES}}$ continues as follows:

1. If there exists a recorded secret key, then $\mathcal{F}_{\text{THRES}}$ forwards $(\mathsf{Decrypt}, c, \mathsf{Both})$ to party P_{1-i}.
   (a) If P_{1-i} replies with allow and $\mathsf{Both} = \phi$, $\mathcal{F}_{\text{THRES}}$ sends $\mathsf{Dec}_{sk}(c)$ only to P_i. Otherwise, if $\mathsf{Both} \neq \phi$, $\mathcal{F}_{\text{THRES}}$ sends $\mathsf{Dec}_{sk}(c)$ to both parties.
   (b) If P_{1-i} replies with $\mathsf{disallow}$, then $\mathcal{F}_{\text{THRES}}$ forwards $\mathsf{disallow}$ to P_i.
2. If there does not exist a recorded secret key, functionality $\mathcal{F}_{\text{THRES}}$ sends $\mathsf{disallow}$ to P_i.

Fig. 2. The (Paillier) Threshold Functionality

4.1 A Distributed Decryption for Paillier

In this section we present a secure decryption protocol in the distributed setting. At first glance it may seem that decryption according to Paillier should follow the same outlines as decryption according to the RSA scheme, where the decrypter raises the ciphertext to the power of the inverse of N modulo $\phi(N)$ as e.g., in [8]. This is because for both schemes the public key is an RSA composite N and the secret key is the factorization of N, and both schemes share some similarities. Therefore, essentially one can apply for Paillier any distributed decryption protocol used for RSA.

In some scenarios, however, this type of algorithm may be problematic. For instance, when decrypting as in RSA (i.e., raising the ciphertext to the inverse of N), the decrypter must extract first the randomness used for computing the ciphertext in order to complete the decryption. As desirable as this property may be, it is problematic in the context of simulation based secure computation because the parties have to present the randomness of the ciphertext instead of proving correctness using ZK proofs. This means that a potential simulator cannot cheat in the decryption. Moreover, recall that Paillier's scheme requires extra computation in order to complete the decryption. This means that on top of raising the ciphertext to the power of the secret value, the outcome must be multiplied with the inverse of the secret key in order to extract the plaintext. In the distributive setting this implies that the parties must keep two types of

shares. When coping with malicious behavior it is not immediately clear how to efficiently verify the parties' computations. Notably, the protocol of Damgård and Jurik [15] circumvents this technicality by having a trusted party picking a secret $d \equiv 1 \bmod N \equiv 0 \bmod \phi(N)$.

Our protocol offers a distributive decryption for Paillier with simulation based security against malicious adversaries without randomness extraction. It is comprised of the following two subprotocols: First, the parties produce multiplicative shares of $\phi(N)^{-1} \in \mathbb{Z}_N$. This protocol is executed only once. Next, the parties run the distributed decryption algorithm using both the additive and multiplicative shares of $\phi(N)$.

Computing Multiplicative Shares of the Secret Key. Note first that the parties can, by local computations, calculate an additive sharing (sk_0^A, sk_1^A) of the private key using values calculated in DKeyGen (cf. Section 3). In the following, they compute an additional set of multiplicative shares (sk_0^M, sk_1^M) for $\phi(N)$, where multiplication is done in $\mathbb{Z}_N$. In order to do that, we use an additional public key encryption scheme π_{DJ}, due to Damgård and Jurik [17], that operates in $\mathbb{Z}^*_{N^2}$, but has an ElGamal flavor. Specifically, a plaintext $m \in \mathbb{Z}_N$ is encrypted by $(g^r, (1+N)^m \cdot h^r \bmod N^2)$ for public key: (N, g, h) such that g is a random square in $\mathbb{Z}^*_N$ and $h = g^r$ for a random r. Note that given $(1+N)^m$, m can be easily computed since discrete logarithm in the subgroup generated by $(1+N)$ is easy. In the following we abuse notation and encrypt elements $c \in \mathbb{Z}^*_{N^2}$ by computing $(g^r, c \cdot h^r \bmod N^2)$. Finally, we note that π_{DJ} enjoys the same advantages of the standard ElGamal scheme, e.g., generating the public key and decrypting distributively.

Protocol 3. [Dsk] *A distributed secret key generation:*

- **Inputs:** *A security parameter 1^n, a public key N and secret shares: p_0, q_0 for P_0 and p_1, q_1 for P_1.*
- **The Protocol**
  1. *The parties run a protocol π_{GEN} for generating a public key $pk_{\text{DJ}} = (g, h)$, and secret key shares for π_{DJ}.*
  2. *Party P_0 sets $sk_0^A = N - p_0 - q_0 + 1$, whereas party P_1 sets $sk_1^A = -(p_1 + q_1)$.*
  3. *Let $N = pk$, then P_0 randomly picks $\delta \in_R \mathbb{Z}^*_N$ and encrypts δ to $c = (c_a, c_b) = (g^{r_c}, (1+N)^{\delta} h^{r_c})$, and sends c to P_1 together with a proof of knowledge for δ via π_{ENC}.*

     Moreover, P_0 computes $c'_0 = (c_a^{sk_0^A} g^{r_0}, c_b^{sk_0^A} h^{r_0})$ and proves that this ciphertext c'_0 and γ_0^4 are consistent using π_{EQ} (where γ_0 is as computed in the biprimality test in Section 3.1). Finally, P_0 records $sk_0^M = \delta$.
  4. *P_1 verifies the proofs π_{ENC} and π_{EQ}, and aborts if they are invalid. Otherwise, P_1 repeats the previous step similarly by computing $c'_1 = (c_a^{sk_1^A} g^{r_1}, c_b^{sk_1^A} h^{r_1})$ sending it to P_0 and proving consistency between c'_1 and γ_1^4.*
  5. *The parties run a distributed decryption to decrypt the multiplication $c'_0 \cdot c'_1$ for P_1. Let $\bar{c}$ denote the result.*
  6. *In case decryption is completed successfully, P_1 records the value $sk_1^M = ((\bar{c} - 1)/N)^{-1}$ where the inverse is computed in $\mathbb{Z}_N$.*

Correctness follows from the following:

$$\left(\frac{c_a^{sk_0^A} \cdot c_b^{sk_1^A} \bmod N^2 - 1}{N}\right)^{-1} = \left(\frac{(1+N)^{(\delta \cdot \phi(N))} - 1}{N}\right)^{-1} = [\delta \cdot \phi(N)]^{-1} \bmod N,$$

Where the last equality follows from the correctness of the decryption. This implies that the parties multiplicatively share $\phi(N)^{-1}$ over $\mathbb{Z}_N$ as required.

Theorem 3. *Assuming hardness of the (composite) DDH and DCR problems, Protocol 3 distributively generates multiplicative shares in the presence of malicious adversaries.*

The complete proof is found in the full version.

A Complete Protocol for Decryption. We assume that the parties are holding additively shares, denoted by $sk_0^{\mathsf{A}}, sk_1^{\mathsf{A}}$, as well as multiplicative shares (modulo the newly generated public key N), denoted by $sk_0^{\mathsf{M}}, sk_1^{\mathsf{M}}$.

Protocol 4. [DDec] *A distributed decryption:*

- **Joint Inputs:** *A security parameter 1^n, a public key N and a ciphertext c to be decrypted for P_0.*
- **Private Inputs:** *A pair of additive shares $sk_0^{\mathsf{A}}, sk_1^{\mathsf{A}}$ (as generated in Section 3.1), and a pair of multiplicative shares $sk_0^{\mathsf{M}}, sk_1^{\mathsf{M}}$ (as generated in Section 4.1) for P_0 and P_1, respectively.*
- **The Protocol:**
  1. *Let $pk = N$, then P_0 begins by randomly picking $\omega \in_R \mathbb{Z}_N$ and $r_\omega \in_R \mathbb{Z}_N^*$, and sending P_1, $c' = \mathsf{Enc}_{pk}(\omega; r_\omega)$ together with a zero-knowledge proof of knowledge for ω via π_{ENC}.*
  2. *The parties run a similar protocol to Protocol 3, where they decrypt $c \cdot c'$. At the end of execution, P_0 records $m = (\tau \cdot sk_0^{\mathsf{M}}) - \omega \bmod N$ for τ the output of P_1 from the decryption protocol.*
  3. *The parties repeat Step 2 as follows. P_0 picks random elements $a, b, r \in \mathbb{Z}_N$ and computes $c'' = (c \cdot c')^a \cdot \mathsf{Enc}(b, r)$. Namely, P_0 computes the encryption of $a \cdot \ell + b$, denoting a one-time MAC for ℓ, where $\ell = \mathsf{Dec}_{sk}(c \cdot c')$.*
     P_0 then proves its computations using a zero-knowledge proof, $\pi_{\text{EXP-RERAND}}$. The parties repeat Step 2 using c'' instead of $c \cdot c'$. Denote by τ' the outcome of P_0.
  4. *In case $\tau' \neq a \cdot \tau + b$, P_0 aborts the execution. Otherwise, it outputs $\tau - \omega$.*
  5. *In case both parties should learn the decryption of c, they repeat this protocol with reversed rolls.*

Theorem 4. *Assuming hardness of the DCR problem, Protocol 4 distributively decrypt according to Paillier, i.e, realizes* DECRYPTION *in $\mathcal{F}_{\text{THRES}}$, with security in the presence of malicious adversaries.*

Intuitively, the security proof follows so that when P_0 is corrupted a simulator $\mathcal{S}$ uses the plaintext received from the trusted party to compute the message sent to P_0. The case that P_1 is corrupted simulation follows easily since P_1 does not learn anything. The complete proof is found in the full version.

5 The Efficiency of Our Protocols

This discussion is split into two parts: a theoretical analysis and a more practical analysis. We remark that all of our zero-knowledge proofs run in constant rounds and require constant number of exponentiations – the only exception is π_{EQ}, employed in our key generation and threshold decryption protocol, for which there is an amortized constant analysis due to Cramer and Damgård [12]. Preliminary results of implementing a protocol for distributively generating an RSA composite in the honest-but-curious setting can be found in [36].

On the Number of Failed Attempts. The complexity of our protocol depends heavily on the number of attempts. Without trial division the protocol has to restart with two freshly generated prime candidates after every rejected composite and the expected number of tests is given by the probability of choosing two random primes. Using the Prime Number Theorem, the expected number of executions: 512 bit primes: $\left(\ln(2^{512})/2\right)^2 \approx 31000$, 1024 bit primes: $\left(\ln(2^{1024})/2\right)^2 \approx 126000$. This can be dramatically improved by employing the trial division test. Following the analysis of [4] it can be shown that the probability a number is a prime, given that it passes the trial division, is computed due to [6] and is,

$$\Pr[p \text{ is prime} \mid p \text{ passes trial division with threshold B}] = 2.57 \cdot \frac{\ln \mathrm{B}}{\mathrm{n}} \left(1 + \mathrm{o}\left(\frac{1}{\mathrm{n}}\right)\right)$$

which for $\ln B = 9$ (i.e., $B = 8103$) and $n = 512$ is approximately $1/22$, and for $n = 1024$ is $1/44$. This means that our protocol needs to test an expected number of 484 candidates when $n = 512$, and 1936 candidates if $n = 1024$. Our protocol is the first to incorporate trial division test securely in the two-party setting, and the above analysis shows how important this is for the efficiency.

Theoretic Efficiency

Key Generation. Ignoring the initial key-setup, the complexity of a single RSA-composite-generation attempt (except the biprimality test) is dominated by the trial divisions; the rest requires constant work and communication. Each trial division requires only constantly many invocations of sub-protocols, including π_{MOD} (and hence of π_{BOUND}) and all these sub-protocols require only a constant number of exponentiations. Thus, the total costs incurred by the entire protocol are linear in the number of trial divisions. Further, all sub-protocols at every step of the full protocol may be run in parallel, hence round complexity is constant.

Biprimality Test. The main part of the biprimality test consists of the verification of the secure exponentiation of the random γ. Further, in the test the parties reach a negligible error probability by repeating the test ℓ times, it must be run e.g. 40 times in order to achieve an error of 2^{-40}. The most expensive part of the test is the execution of π_{EQ} as it is a cut and choose protocol, i.e. we need $O(\ell)$ exponentiations overall for each run, where ℓ is some statistical security

parameter. However, as noted above this may be brought down to amortized constant overhead using the techniques of Cramer and Damgård [12]. Further, as the ℓ tests may be run in parallel, round complexity is constant as well.

Secret-Key Shares. The generation of the multiplicative key shares requires an invocation of π_{EQ}, hence complexity is not constant due to the use of cut and choose. However, this is a one-time execution, hence the cost can be amortized over an arbitrary number of subsequent decryptions.

Threshold Decryption. This protocol is dominated by the invocation of π_{EQ}. In a real-world scenario we expect to decrypt a large number of cipher texts. The amortized cost of this can be improved to constant using [12].

Practical Considerations. To ease the security proof, we have so far taken the approach of applying ZK-proofs at all stages in order to catch cheaters immediately. However, a more optimistic approach allows for a more efficient protocol: All but one of our RSA-composite-generation attempts fail, and most of the ZK-proofs are only needed for the successful modulus generation – hence they may be postponed. In addition to this, further optimizations for distributed RSA key generation are possible. We refer to Boneh and Franklin [4] for a list of general optimizations some of which are also applicable in our setting.

For the failing RSA-composite-generation attempts, we utilize the fact that the encryptions provided can be viewed as binding commitments. On failure, the parties reveal all random choices, thereby allowing the other party to verify their correct behavior by checking the correctness of the other party's messages. Thus, overall efficiency of the many *failing* attempts will not be much more costly than twice that of failing attempts for the passively secure protocol. Once an attempt succeeds, ZK-proofs are used to ensure that this was correct. Slightly more formally, the key idea is that the simulator must know that the adversary is cheating (and that an honest party would detect this later, i.e. that the invocation should fail). We cannot simply postpone *all* proofs; care must be taken to allow simulation and to not reveal information that would allow a malicious party to, e.g., fake some zero-knowledge proof at a later point.

Generating the prime candidate. We may omit the invocations of π_{BOUND} on p_i and q_i, as this statement is implicitly shown by the invocations of π_{MOD} in the trial divisions. Further, verification may be postponed until we believe we have successfully generated an RSA-modulus; we cannot ensure correctness underway, but the encryption will be the same, thus, we still accept or reject as if we had run π_{BOUND} immediately.

Trial division. We may postpone the invocations of π_{MOD} at the cost of a few extra executions of simple proofs of knowledge, such as π_{ENC}. This ensures that the party knows its input, *and* that the simulator knows whether a later invocation of π_{MOD} may be successful. If a trial division fails incorrectly, the honest party learns this when the corrupt party reveals its random choice, including its share of the random prime. If a trial division succeeds incorrectly, this can

be discovered easily by performing local trial-division on N, we may even use a *larger* bound for the trial division as this can be performed very efficiently on the public N. The only remaining possibility is the case where the test should succeed, and did so despite one party providing an incorrect input. This case is handled by executing π_{MOD} for each trial division once the biprimality test succeeds, at which point the honest party will detect the incorrect behavior.

Computing and verifying the product. For the Paillier computation, we may postpone all checks except the proof that $\tilde{N}$ was the plaintext of the encryption supplied by P_1. Privacy of P_0 follows from the semantic security of Paillier encryption, while privacy for P_1 follows from the fact the encryption sent back by P_1 only contains the desired result. Leakage from constructing a potentially incorrect value is eliminated by the eventual execution of the full ZK-proofs. Alternatively, we may avoid verifying the product altogether. This may leak *a single bit of information*, namely whether some function on the shares of the honest party equals the still hidden modulus, N. Depending on the setting, this leakage may or may not be acceptable.

Biprimality test. The invocation of π_{EQ} can be postponed. If the test fails, the parties simply reveal their shares of the candidates, and check for honest behavior. On the other hand, if the test succeeds, the parties have either determined an RSA composite or one of them has cheated. They now execute π_{EQ} to determine which of the two is the case. Since the simulator knows the shares of the corrupt party, it is straightforward for it to check if the value supplied is the correct one.

References

1. Algesheimer, J., Camenisch, J., Shoup, V.: Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 417–432. Springer, Heidelberg (2002)
2. Baudron, O., Fouque, P.A., Pointcheval, D., Poupard, G., Stern, J.: Practical multi-candidate election system. In: PODC, pp. 274–283. ACM Press (2001)
3. Blackburn, S., Blake-Wilson, S., Burmester, M., Galbraith, S.: Shared generation of shared RSA keys, `http://cacr.math.uwaterloo.ca/techreports/1998/corr98-19.ps`
4. Boneh, D., Franklin, M.K.: Efficient generation of shared RSA keys. J. ACM 48(4), 702–722 (2001)
5. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
6. De Bruijn, N.: On the number of uncanceled elements in the sieve of eratosthenes. Proc. Neder. Akad. Wetensh. 53, 803–812; Reviewed in LeVeque Reviews in Number Theory 4(28), 221
7. Camenisch, J., Kiayias, A., Yung, M.: On the Portability of Generalized Schnorr Proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009)

8. Catalano, D., Gennaro, R., Howgrave-Graham, N., Nguyen, P.Q.: Paillier's cryptosystem revisited. In: ACM Conference on Computer and Communications Security, pp. 206–214 (2001)
9. 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)
10. Cocks, C.: Split Generation of RSA Parameters with Multiple Participants. In: Darnell, M.J. (ed.) IMACC 1997. LNCS, vol. 1355, pp. 200–212. Springer, Heidelberg (1997)
11. Coppersmith, D.: Small Exponents to Polynomial Equations, and Low Exponent RSA Vulnerabilities. Journal of Cryptology 10, 233–260 (1997)
12. Cramer, R., Damgård, I.: On the Amortized Complexity of Zero-Knowledge Protocols. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 177–191. Springer, Heidelberg (2009)
13. Cramer, R., Damgård, I.B., Nielsen, J.B.: Multiparty Computation from Threshold Homomorphic Encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (2001)
14. 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)
15. Damgård, I., Jurik, M.: A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
16. Damgård, I., Jurik, M.: Client/Server Tradeoffs for Online Elections. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 125–140. Springer, Heidelberg (2002)
17. 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)
18. Damgård, I., Mikkelsen, G.L.: Efficient, Robust and Constant-Round Distributed RSA Key Generation. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 183–200. Springer, Heidelberg (2010)
19. Damgård, I., Nielsen, J.B.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
20. Damgård, I., Nielsen, J.B.: Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
21. Desmedt, Y.G.: Threshold cryptography. European Transactions on Telecommunications 5(4), 449–457 (1994)
22. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Trans. Info. Theory IT 31, 469–472 (1985)
23. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. J. Cryptology 1(2), 77–94 (1988)
24. 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)
25. Fouque, P.A., Poupard, G., Stern, J.: Sharing Decryption in the Context of Voting or Lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
26. Frankel, Y., Mackenzie, P.D., Yung, M.: Robust efficient distributed RSA-key generation. In: STOC 1998, pp. 663–672. ACM Press (1998)

27. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust Threshold DSS Signatures. Information and Computation 164(1), 54–84 (2001)
28. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. Journal of Cryptology 20(1), 51–83 (2007)
29. Gennaro, R., Krawczyk, H., Rabin, T.: Robust and Efficient Sharing of RSA Functions. Journal of Cryptology 13(2), 273–300 (2000)
30. Gilboa, N.: Two Party RSA Key Generation. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 116–129. Springer, Heidelberg (1999)
31. Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T.: Efficient RSA key generation and threshold Paillier in the two-party setting. Cryptology ePrint Archive, Report 2011/494 (2011)
32. Hazay, C., Toft, T.: Computationally Secure Pattern Matching in the Presence of Malicious Adversaries. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 195–212. Springer, Heidelberg (2010)
33. Jarecki, S., Liu, X.: Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)
34. Jarecki, S., Shmatikov, V.: Efficient Two-Party Secure Computation on Committed Inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
35. Lipmaa, H.: On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
36. Nicolosi, A.A.: Efficient RSA Key Generation Protocol in a Two-Party Setting and its Application into the Secure Multiparty Computation Environment – Master Thesis. Department of Computer Science Aarhus University, Denmark (2011)
37. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
38. Poupard, G., Stern, J.: Generation of Shared RSA Keys by Two Parties. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 11–24. Springer, Heidelberg (1998)
39. 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)
40. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4, 161–174 (1991)
41. Shoup, V.: Practical Threshold Signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)

A Building Blocks: Zero-Knowledge Proofs

The Zero-Knowledge Proofs are formally defined and described in detail in the full version.

Discrete Logarithms

1. Protocol π_{DL} which demonstrates knowledge of a discrete logarithm [40]. $\mathcal{R}_{\text{DL}} = \{((\mathbb{G}, g, h), w) \mid h = g^w\}$.
2. Protocol π_{DH} which demonstrates that a quadruple (g_0, g_1, h_0, h_1) [9]. $\mathcal{R}_{\text{DH}} = \{((\mathbb{G}, g_0, g_1, h_0, h_1)\, w) \mid h_i = g_i^w \text{ for } i \in \{0, 1\}\}$.

Public Key Cryptosystems (and Commitment Schemes)

1. Protocol π_{ENC} demonstrates knowledge of the plaintext of an encryption. $\mathcal{R}_{\text{ENC}} = \{((c, pk), (\alpha, r)) \mid c = \mathsf{Enc}_{pk}(\alpha; r)\}$. Protocols due to Schnorr [40] (ElGamal) and Cramer et al. [13] (Paillier).
2. Protocol π_{ZERO} demonstrates that a ciphertext c is an encryption of zero. $\mathrm{L}_{\text{ZERO}} = \{((c, pk), r) \mid c = \mathsf{Enc}_{pk}(0; r)\}$. For ElGamal this is merely π_{DH}. For Paillier encryption this is a proof of N'th power shown by [15].
3. The zero-knowledge proof of knowledge π_{MULT} proves that the plaintext of c_2 is the product of the two plaintexts encrypted by c_0, c_1. $\mathcal{R}_{\text{MULT}} = \{((c_0, c_1, c_2, pk), (\alpha, r_\alpha, r_0)) \mid c_1 = \mathsf{Enc}_{pk}(\alpha; r_\alpha) \wedge c_2 = c_0^\alpha \cdot \mathsf{Enc}_{pk}(0; r_0)\}$; This proof due to Damgård and Jurik [15] (for both Paillier and ElGamal). A similar proof of knowledge is possible when the contents of *all three* commitments are known, [19,20]; this is required in π_{BOUND} below.
4. Protocol π_{BOUND} demonstrates boundedness of an encrypted value, i.e. that the plaintext is smaller than some public threshold B. Formally, $\mathrm{L}_{\text{BOUND}} = \{((c, pk, B), (\alpha, r)) \mid c = \mathsf{Enc}_{pk}(\alpha; r) \wedge \alpha < B \in \mathbb{N}\}$. The "classic" solution is to provide encryptions to the individual bits and prove in zero-knowledge that they are bits using the compound proof of Cramer et al. [14]. The actual encryption is then constructed from these. An alternative is to take a detour around integer commitments; this allows a solution requiring only $O(1)$ exponentiations [5,35,16].
5. The proof π_{EQ} is of correct exponentiation in the group $\mathbb{G}'$ with encrypted exponent (where the encryption scheme does *not* utilize the description of $\mathbb{G}'$). Formally, $\mathcal{R}_{\text{EQ}} = \{((c, pk, \mathbb{G}', h, h'), (\alpha, r)) \mid \alpha \in \mathbb{N} \wedge c = \mathsf{Enc}_{pk}(\alpha; r) \wedge h, h' \in \mathbb{G}' \wedge h' = h^\alpha\}$. The protocol uses a simple cut and choose approach, and originates from [7].

New Zero-Knowledge Proofs

1. We include the folklore protocol π_{RSA} for proving that N and $\phi(N)$ are co-prime for some integer N, i.e. the protocols demonstrate membership of the language, $\mathrm{L}_{\text{RSA}} = \{(N, (\text{factorization of } N)) \mid N \in \mathbb{N} \wedge \text{GCD}(N, \phi(N)) = 1\}$.
2. We also require a zero-knowledge proof, π_{MOD}, for proving consistency between two ciphertexts in the sense that one plaintext is the other one reduced modulo a public, fixed value (prime). This is required for proving correctness within the trial division stage included in the key generation protocol (cf. Section 3). Formally, $\mathrm{L}_{\text{MOD}} = \{((c, c', p, pk), (\alpha, r, r')) \mid c = \mathsf{Enc}_{pk}(\alpha; r) \wedge c' = \mathsf{Enc}_{pk}(\alpha \bmod p; r')\}$.
3. For public Paillier key N, we require Σ-protocol $\pi_{\text{EXP-RERAND}}$ that allows a prover to demonstrate that ciphertext c' is in the image of $\phi : \mathbb{Z}_N \times \mathbb{Z}_{N^2}^* \mapsto \mathbb{Z}_{N^2}^*$, defined by $\phi(a, r) = c^a \cdot r^N \bmod N^2$ for a fixed ciphertext $c \in \mathbb{Z}_{N^2}^*$. Namely, $\mathrm{L}_{\text{EXP-RERAND}} = \left\{((N, c, c'), (\alpha, r)) \mid c' = c^\alpha \cdot r^N\right\}$.

A Zero-Knowledge Proof for $\boldsymbol{\pi_{\text{VERLIN}}}$. In this section we give the details of ZK proof π_{VERLIN} used in Step 3a of Protocol 1. Let N_P be a public Paillier key, with $g = N_P + 1$ generating the plaintext ring. π_{VERLIN} is a Σ-protocol allowing a prover P to demonstrate to a verifier V that a Paillier ciphertext, c_x has been computed based on two other ciphertexts c and c' as well as a known value, i.e. that P knows a preimage of c_x with respect to $\phi_{(c,c')}(x, x', x'', r_x) = c^x \cdot c'^{x'} \cdot \mathsf{Enc}(x'', r_x)$. This is done by first picking a, a', a'' uniformly at random from $\mathbb{Z}_{N_P}$ and r_a uniformly at random from $\mathbb{Z}^*_{N_P}$, and sending $c_a = \phi_{(c,c')}(a, a', a'', r_a)$ to the verifier, V. V then picks a uniformly random t-bit challenge e, and sends this to P, who replies with the tuple $(z, z', z'', r_z) = (xe + a, x'e + a', x''e + a'', r_x^e r_a)$. V accepts if and only if $\phi_{(c,c')}(z, z', z'', r_z) = c_x^e \cdot c_a$.

Plaintext-Checkable Encryption

Sébastien Canard[1], Georg Fuchsbauer[2],
Aline Gouget[3], and Fabien Laguillaumie[4]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] University of Bristol, Dept. Computer Science, UK
[3] Gemalto, Security Lab, Meudon, France
[4] UCBN and CNRS/ENSL/INRIA/UCBL LIP, Lyon, France

Abstract. We study the problem of searching on encrypted data, where the search is performed using a plaintext message or a keyword, rather than a message-specific trapdoor as done by state-of-the-art schemes. The use cases include delegation of key-word search e.g. to a cloud data storage provider or to an email server, using a plaintext message. We define a new cryptographic primitive called *plaintext-checkable encryption* (PCE), which extends public-key encryption by the following functionality: given a plaintext, a ciphertext and a public key, it is universally possible to check whether the ciphertext encrypts the plaintext under the key. We provide efficient generic random-oracle constructions for PCE based on any probabilistic or deterministic encryption scheme; we also give a practical construction in the standard model. As another application we show how PCE can be used to improve the efficiency in group signatures with *verifier-local revocation* (VLR) and backward unlinkability. These group signatures provide efficient revocation of group members, which is a key issue in practical applications.

Keywords: Deterministic/probabilistic encryption, unlinkability, group signature with VLR and backward unlinkability.

1 Introduction

The problem of searching on data that is encrypted has been studied intensively and in many different scenarios. For instance, the problem of delegation of keyword search on private databases to a data storage provider concerns users who upload their data to a provider they do not fully trust. When the user wants to delegate keyword search on his own encrypted data to the provider, he usually has to transmit a corresponding message-dependent trapdoor (or encrypted keyword) which enables the provider to perform the search. When the databases are public, the user wishes to delegate the search on public data to a cloud data storage provider without revealing the plaintext content of the search. Another setting is the delegation of search to an email gateway [9], where data collected by the mail server is from third parties (contrary to the private-key setting as above) and the database is not public.

Most of the constructions proposed in the literature are based either on symmetric-key cryptography to encrypt the plaintext message or keyword, or

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 332–348, 2012.

on searchable encryption without the ability to decrypt the message as done in [22]. The security of the search process in the state of the art of public-key encryption constructions has always been studied assuming that the search process uses a secret trapdoor and not a plaintext message. In this work we focus on this latter case, which is naturally related to public-key cryptography. This case can in practice be very useful when the database contains relations between different words (a name and a status for example) and it is these relations that have to be kept secret rather than the words themselves. Thus, when searching e.g. the number of persons having the status "important illness", the keyword "important illness" is not secret and can be directly used to perform the search. Many functionalities extending the basic setting of public-key encryption have been considered, in particular related to data search. For example, *decryptable searchable encryption* [13] allows someone having a trapdoor corresponding to a message, to test whether a given ciphertext encrypts this message. Another example is *encryption with equality test*, proposed in [23]. Using the equality test, one can check whether two ciphertexts encrypt the same plaintext.

In this paper we propose and study a new cryptographic primitive we call *plaintext-checkable encryption* (PCE). A plaintext-checkable encryption scheme is a probabilistic public-key encryption scheme with the additional functionality that anyone can test whether a ciphertext c is the encryption of a given plaintext message m under a public encryption key pk. Despite this functionality, we demand that the ciphertext leak as little information as possible about the plaintext. Of course, a PCE scheme cannot achieve the standard notion of *indistinguishability under chosen-plaintext attack*, as an adversary choosing two messages and receiving the encryption of one of them can simply test which message was encrypted. The same holds when the encryption algorithm is *deterministic*: an adversary can just re-encrypt candidate messages and thus break classical indistinguishability.

As was done in the case of deterministic encryption [3], we assume that the plaintexts are drawn from a space of large min-entropy; indistinguishability means thus the impossibility of distinguishing ciphertexts of messages drawn from different high min-entropy spaces. We show however that we can achieve a strictly stronger security notion than indistinguishability for deterministic encryption [3,4]: an adversary is not able to distinguish two encryptions of the same message from encryptions of different messages. This notion cannot be achieved by deterministic encryption, since there is only one possible ciphertext per message, and encryption with equality check cannot achieve it either. We say that an encryption scheme satisfies *unlinkability* if no polynomial-time adversary can win the following game: a challenger draws two messages from a high min-entropy space of the adversary's choice and gives the adversary either encryptions of the two messages or two encryptions of one message, and the adversary has to decide which is the case. We relate this notion to the different types of indistinguishability, showing e.g. that it is strictly stronger than the indistinguishability notion for deterministic encryption, and we argue that our notion is sufficient for our applications. We provide efficient generic constructions of PCE schemes

satisfying unlinkability based either on probabilistic or deterministic encryption with a security proof in the random-oracle model (ROM).[1] We also build a practical construction based on ElGamal encryption, secure in the standard model.

Apart from its immediate applications to searching on encrypted data, PCE lends itself naturally to improve the efficiency of group signatures with *verifier-local revocation* (VLR). Group signatures allow members of a group to sign on behalf of the group without revealing their individual identity. Group signatures with VLR were introduced by Boneh and Shacham [10] and allow efficient revocation of group members, which is a key issue in practical applications. In VLR group signatures the revocation messages only have to be sent to signature verifiers, as opposed to both signers and verifiers in previous schemes. We note that unlinkability of ciphertexts is precisely the property required by the encryptions contained in group signatures [8,11]. We show that PCE can be used to encrypt a user-specific revocation token, like a certificate, which will be part of a group signature. A group member can then be revoked by publishing the token, as every verifier can apply the plaintext check to the encrypted token in order to determine whether it corresponds to a revoked user. Since tokens will be drawn from a high min-entropy space, two group signatures containing the same token are unlinkable by the security of the PCE. Our VLR group signature scheme achieves *backward unlinkability* and is proven secure in the standard model.

The paper is organized as follows. In Sect. 2 we formally define plaintext-checkable encryption and we give security definitions and compare them to existing security notions for public-key encryption. In Sect. 3 we provide generic constructions of PCE in the random-oracle model based on either deterministic or probabilistic encryption, while Sect. 4 gives the description of our practical construction in the standard model. We finally show in Sect. 5 how PCE can be used to design very practical group signatures with VLR. Due to space limitations, proofs are omitted but are available in the full version.

2 Plaintext-Checkable Encryption

We define here the notion of plaintext-checkable encryption and its security.

2.1 Definition of Plaintext-Checkable Encryption

Let $k \in \mathbb{N}$ be a security parameter. A *plaintext-checkable encryption scheme* (PCE for short) is composed of the following algorithms (of which the first 3 constitute a public-key encryption scheme).

- KeyGen is a probabilistic algorithm which takes as input 1^k and outputs a key pair (pk, sk) of public and secret key, respectively.
- $\mathsf{Encrypt}$ is a probabilistic algorithm which takes as inputs 1^k, a public key pk and a plaintext $m \in \{0,1\}^*$ and outputs a ciphertext c.

[1] It may be possible to design PCE schemes from any decryptable searchable encryption scheme by simply publishing trapdoors (one trapdoor per message or, in some cases, the master trapdoor). However, our constructions are more efficient.

- $\mathsf{Decrypt}$ is a deterministic algorithm which takes as inputs 1^k, a ciphertext c and a secret key sk and outputs either a plaintext m or $\perp$.
- PCheck is a deterministic algorithm which takes as inputs 1^k, a ciphertext c, a public key pk and a putative message m. It outputs 1 if c is an encryption of m, and 0 otherwise.

These algorithms must verify the following properties of *correctness*.

Correctness of decryption: $\forall k \in \mathbb{N}$ and $m \in \{0,1\}^*$,
$\Pr\left[(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k), c \xleftarrow{\$} \mathsf{Encrypt}(1^k, pk, m) : \mathsf{Decrypt}(1^k, sk, c) = m\right] = 1.$

Correctness of plaintext check (perfect consistency): $\forall k \in \mathbb{N}$ and $m \in \{0,1\}^*$,

$$\Pr\Big[(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k), c \xleftarrow{\$} \mathsf{Encrypt}(1^k, pk, m) : \mathsf{PCheck}(1^k, c, pk, m) = 1\Big] = 1.$$

The property of perfect consistency is implied by the correctness of decryption and the two following properties, which guarantee that PCheck behaves as expected. The following two notions state that if a ciphertext decrypts to a plaintext then PCheck matches them (completeness) and if PCheck matches a ciphertext to a plaintext then the former encrypts the latter (soundness).

Checking completeness: no adversary is able to output a ciphertext c which decrypts to a message that is refused by PCheck on input c. Formally, for every $k \in \mathbb{N}$ and every probabilistic polynomial-time (p.p.t.) algorithm $\mathcal{A}$ that, on inputs 1^k and a public key pk, outputs a ciphertext c, the following probability should be negligible:

$$\Pr\Big[(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k), c \xleftarrow{\$} \mathcal{A}(1^k, pk), m \xleftarrow{\$} \mathsf{Decrypt}(1^k, c, sk) \ : \ \mathsf{PCheck}(1^k, pk, c, m) = 0\Big] \ .$$

Checking soundness: this property states that no adversary should be able to produce a plaintext and ciphertext such that the decryption and the check procedures do not agree on the plaintext related to c. More formally, for every $k \in \mathbb{N}$ and every p.p.t. algorithm $\mathcal{A}$ that, on inputs 1^k and a public key pk, outputs a ciphertext c and a plaintext $\tilde{m}$, the following probability should be negligible:

$$\Pr\Big[(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k), (c, \tilde{m}) \xleftarrow{\$} \mathcal{A}(1^k, pk), m \xleftarrow{\$} \mathsf{Decrypt}(1^k, c, sk) \ : \ m \neq \tilde{m} \wedge \mathsf{PCheck}(1^k, pk, c, \tilde{m}) = 1\Big] \ .$$

2.2 A Taxonomy of Indistinguishability

The classical property of indistinguishability (for public-key encryption schemes) cannot be achieved by a PCE due to the ability to check the plaintext messages

$\underline{\mathbf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(k)}$

$b \xleftarrow{\$} \{0,1\}$
$(pk, sk) \leftarrow \mathcal{G}(1^k)$
$(m_0, m_1, st) \leftarrow \mathcal{A}_f(1^k, pk)$
$c \leftarrow \mathcal{E}(1^k, pk, m_b)$
$b' \leftarrow \mathcal{A}_g(1^k, c, st)$
Return $(b' = b)$

$\underline{\mathbf{Exp}^{\mathsf{unlink}}_{\Pi,\mathcal{A}}(k)}$

$b \xleftarrow{\$} \{0,1\}$
$(pk, sk) \leftarrow \mathcal{G}(1^k)$
$m_0 \leftarrow \mathcal{A}_f(1^k, pk)$
$m_1 \leftarrow \mathcal{A}_f(1^k, pk)$
$c_0 \leftarrow \mathcal{E}(1^k, pk, m_b)$
$c_1 \leftarrow \mathcal{E}(1^k, pk, m_1)$
$b' \leftarrow \mathcal{A}_g(1^k, pk, c_0, c_1)$
Return $(b' = b)$

$\underline{\mathbf{Exp}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k)}$

$b \xleftarrow{\$} \{0,1\}$
$m \leftarrow \mathcal{A}_f(1^k, b)$
$(pk, sk) \leftarrow \mathcal{G}(1^k)$
$c \leftarrow \mathcal{E}(1^k, pk, m)$
$b' \leftarrow \mathcal{A}_g(1^k, pk, c)$
Return $(b' = b)$

Fig. 1. Security experiments for indistinguishability of Π

(see below). We discuss in this section the properties of indistinguishability for encryption schemes.

In the following, we denote by $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ a secure encryption scheme. Depending on the context, Π can be either probabilistic (denoted Π_p) or deterministic (denoted Π_d). We first remark that a PCE can also be represented as an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D}) = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, in the notation from Sect. 2.1.

An adversary $\mathcal{A}$ is defined by a pair of algorithms denoted by $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$, representing the find and guess stage of the experiment, respectively. The adversary $\mathcal{A}$ is said to be polynomial if each constituent algorithm has a running time polynomial in its input length. It is assumed that $\mathcal{A}_f$ and $\mathcal{A}_g$ share neither coins nor state. We study three security experiments for the indistinguishability properties of an encryption scheme Π; the three security experiments, denoted by $\mathbf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(k)$, $\mathbf{Exp}^{\mathsf{unlink}}_{\Pi,\mathcal{A}}(k)$ and $\mathbf{Exp}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k)$, are described in Fig. 1. We first define two classes of adversaries.

Definition 1 (High min-entropy). *An adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ is* legitimate *if there exists a function $\ell(\cdot)$ s.t. for all c and all $m \in [\mathcal{A}_f(1^k, c)]$ we have $|m| = \ell(k)$ (where c can be a bit, as for ind-det adversaries, or a public key, as for ind-cpa and unlink adversaries).*

Moreover, we say that an adversary $\mathcal{A} = (\mathcal{A}_f, A_g)$ has min-entropy μ *if*

$$\forall k \in \mathbb{N}\ \forall c\ \forall m : \Pr\left[m' \leftarrow \mathcal{A}_f(1^k, b)\ :\ m' = m\right]\ \leq\ 2^{-\mu(k)}\ .$$

$\mathcal{A}$ is said to have high min-entropy *if it has min-entropy μ with $\mu(k) \in \omega(\log k)$.*

The first experiment $\mathbf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(k)$ represents the standard indistinguishability property for probabilistic encryption schemes.

Definition 2 (IND-CPA). *Let $k \in \mathbb{N}$, let $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let $\boldsymbol{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}$ be as defined in Fig. 1 and denote $\boldsymbol{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(k) := 2 \cdot \Pr\left[\boldsymbol{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi,\mathcal{A}}(k) \rightarrow \mathsf{true}\right] - 1$. We say that Π* satisfies indistinguishability under a chosen-plaintext attack *if for every legitimate p.p.t. adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$, the advantage $\boldsymbol{Adv}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k)$ is negligible.*

The experiment $\mathbf{Exp}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k)$ is a simplified definition of the indistinguishability property for deterministic encryption introduced in [4], which has been shown to be equivalent to the original definition considered in [3]. We simplify the original definition by considering adversaries that produce distributions of *messages* rather than distributions of message vectors[2].

Definition 3 (IND-DET [4]). *Let $k \in \mathbb{N}$, let $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, and let $\boldsymbol{Exp}^{\mathsf{in\text{-}det}}_{\Pi,\mathcal{A}}$ be as defined in Fig. 1. Let $\boldsymbol{Adv}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k) := 2 \cdot \Pr\left[\boldsymbol{Exp}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k) \to \mathsf{true}\right] - 1$. We say Π* satisfies ind-det *if for every legitimate p.p.t. adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ with high min-entropy, $\boldsymbol{Adv}^{\mathsf{ind\text{-}det}}_{\Pi,\mathcal{A}}(k)$ is negligible.*

We define the third security experiment as the infeasibility of deciding whether two ciphertexts encrypt the same message. The definition shares with ind-det that the messages have to be chosen from a high min-entropy space: otherwise the notion is not satisfiable by a plaintext-checkable scheme, since the adversary could simply check all messages. As we will show all along this paper, this security definition is achievable by plaintext-checkable schemes and sufficient for our applications.

Definition 4 (UNLINK). *Let $k \in \mathbb{N}$ and $\boldsymbol{Adv}^{\mathsf{unlink}}_{\Pi,\mathcal{A}}(k) := 2 \cdot \Pr\left[\boldsymbol{Exp}^{\mathsf{unlink}}_{\Pi,\mathcal{A}}(k) \to \mathsf{true}\right] - 1$, for an encryption scheme $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ with $\boldsymbol{Exp}^{\mathsf{unlink}}_{\Pi,\mathcal{A}}$ as defined in Fig. 1. We say Π* has unlinkable encryptions *(or "satisfies unlink") if for every legitimate p.p.t. adversary $\mathcal{A}$ with high min-entropy, $\boldsymbol{Adv}^{\mathsf{unlink}}_{\Pi,\mathcal{A}}(k)$ is negligible.*

We now give a complete taxonomy of all these security notions and we prove (see the full version) that the unlink notion falls strictly between ind-cpa of probabilistic encryption, and ind-det of deterministic encryption. More precisely, we show the following relation:

$$\textbf{IND-CPA} \subsetneq \textbf{UNLINK} \subsetneq \textbf{IND-DET}.$$

This means that every scheme that achieves ind-cpa is unlink and every scheme that achieves unlink is ind-det. On the other hand, there are schemes that are unlink but not ind-cpa, and others satisfying ind-det but not unlink.

It is obvious that a PCE scheme cannot be ind-cpa since the adversary could forward m_0 and m_1 as st from $\mathcal{A}_f$ to $\mathcal{A}_g$, which could then apply PCheck to the challenge c and for example m_0, and win the experiment with overwhelming probability. As a consequence, the somewhat best we can hope for in the case of PCE is unlinkability. We will thus show that our schemes satisfy this new security notion.

Deterministic encryption schemes [3], though trivially plaintext-checkable, cannot satisfy the property unlink since every two encryptions of a message are equal, which allows a trivial check of plaintext equality. One may attempt to construct plaintext-checkable encryption from an encryption scheme with equality

[2] The original definition considers vectors of messages since (unlike for ind-cpa-secure encryption) there is no reduction to the single-message case by a hybrid argument for deterministic encryption.

test as described in [23] by simply encrypting the message and then performing the test of equality. However, this scheme does not satisfy unlink either for obvious reasons. Moreover, as noticed by the authors, their Test function only works properly when the ciphertexts are two real encryptions of messages, as this procedure does not check the validity of the ciphertexts.

It thus remains open to give a construction (practical or generic) with the above features, namely providing a PCheck procedure, while maintaining unlinkability. We give such constructions in the two following sections.

3 Generic Constructions for PCE in the ROM

We show how to obtain secure PCE schemes using a secure probabilistic or deterministic encryption scheme with security proofs in the random-oracle model.

3.1 A PCE Based on a Probabilistic Encryption Scheme

In this construction a message m is encrypted by first choosing a random string r and computing a hash value ρ of the message and r. This value ρ is then used as the random coins of the probabilistic encryption algorithm to encrypt m, and r is added to the ciphertext. The algorithm PCheck consists essentially in re-computing ρ and then re-encrypting the message with random coins ρ and comparing it to the candidate ciphertext. Our solution is described in Fig. 2. The triple $\Pi_p = (\mathcal{G}_p, \mathcal{E}_p, \mathcal{D}_p)$ denotes a probabilistic encryption scheme satisfying indistinguishability under chosen-message attack, and $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{\ell(k)}$ denotes a hash function modeled as a random oracle.

The following theorem states the security of the construction of Fig. 2, i.e. that it satisfies unlinkability. Essentially, the unlinkability of this construction

Algorithm $\mathsf{KeyGen}(1^k)$
$(\overline{pk}, \overline{sk}) \xleftarrow{\$} \Pi_p.\mathcal{G}_p(1^k)$
$pk \leftarrow \overline{pk}$
$sk \leftarrow \overline{sk}$
return (pk, sk)

Algorithm $\mathsf{Encrypt}(1^k, pk, m)$
$\overline{pk} \leftarrow pk$
$r \xleftarrow{\$} \{0,1\}^{\ell(k)}$
$\rho \leftarrow \mathcal{H}(m\|r)$
$\overline{c} \leftarrow \Pi_p.\mathcal{E}_p(1^k, \overline{pk}, m; \rho)$
$C \leftarrow (\overline{c}, r)$
return C

Algorithm $\mathsf{Decrypt}(1^k, sk, C)$
$(\overline{c}, r) \leftarrow C$
$\overline{sk} \leftarrow sk$
$m \leftarrow \Pi_p.\mathcal{D}_p(1^k, \overline{sk}, \overline{c})$
return m

Algorithm $\mathsf{PCheck}(1^k, pk, C, m)$
$(\overline{c}, r) \leftarrow C$
$\overline{pk} \leftarrow pk$
$\rho \leftarrow \mathcal{H}(m\|r)$
$\tilde{c} \leftarrow \Pi_p.\mathcal{E}_p(1^k, \overline{pk}, m; \rho)$
if $\tilde{c} = \overline{c}$ then return 1
else return 0

Fig. 2. Unlinkable PCE from an ind-cpa encryption scheme Π_p

follows from the indistinguishability of the underlying encryption scheme. However, quite some care needs to be taken to ensure that the simulation in the reduction is perfect, as the adversary against unlinkability may make queries to the random oracle that the simulator cannot answer.

Theorem 1. *If Π_p satisfies* ind-cpa *then the PCE from Fig. 2 satisfies* unlink.

Proof (sketch, see full version for the full proof). We show that a successful adversary $\mathcal{A}$ against unlink of our PCE scheme can be used to construct an adversary $\mathcal{B}$ against ind-cpa of Π_p. A natural construction of $\mathcal{B}$ is the following: $\mathcal{B}_f$ runs $\mathcal{A}_f$ twice and outputs the obtained messages m_0 and m_1. The challenger then gives $\mathcal{B}_g$ a Π_p-encryption c of m_b. Now $\mathcal{B}_g$ must use $\mathcal{A}_g$ to determine b. Playing the unlinkability game, $\mathcal{A}_g$ expects two PCE ciphertexts; one of m_b and one of m_1. While the latter can be computed honestly, $\mathcal{B}_g$ could construct the former as (c, r_0), for some random r_0.

However, this implicitly defines $\mathcal{H}(m_b, r_0)$ to be the randomness $\mathcal{B}$'s challenger used in constructing c; $\mathcal{B}$ can thus not answer this random-oracle query and the simulation might fail. In a series of lemmas, we show that under ind-cpa of Π_p, the probability of $\mathcal{A}_g$ (who does not know m_0 and m_1) querying m_0 or m_1 to $\mathcal{H}$ is negligible. We first show that this holds if $\mathcal{B}$'s challenger's bit $b = 0$:

Suppose in game unlink when $b = 0$, $\mathcal{A}_g$ queries $(m_0 \| r)$ (for some r) to the random oracle $\mathcal{H}$. Then we construct $\mathcal{B}'$ that breaks ind-cpa. It uses $\mathcal{A}_f$ to sample m_0 and m_1, gets an encryption c of m_d from its challenger and then runs $\mathcal{A}_g$ on (c, r_0) (for some random r_0) and a PCE encryption of an independent message m'. Since $\mathcal{A}_g$ does not have any information on m_{1-d} (which was sampled from a high min-entropy space), querying e.g. m_0 must mean $d = 0$. Thus if $\mathcal{A}_g$ makes a query to $\mathcal{H}$ containing m_d, $\mathcal{B}'_g$ outputs d as its guess. Note that the issue of correctly simulating the random oracle does not arise here, as $\mathcal{B}'_g$ aborts as soon as $\mathcal{A}_g$ makes a critical query. Analogously, we show that when $b = 0$, the probability that $\mathcal{A}_g$ queries $(m_1 \| \cdot)$ is negligible.

It remains to prove that when $b = 1$ then $\mathcal{A}_g$ queries $(m_1 \| \cdot)$ with negligible probability. Again, assuming $\mathcal{A}_g$ makes such a query, we construct $\mathcal{B}''$ breaking ind-cpa. As before, $\mathcal{B}''$ uses $\mathcal{A}_f$ to sample m_0 and m_1 and receives c. Now $\mathcal{B}''_g$ picks a random bit d and sends $\mathcal{A}_g$ the following: (c, r_0), for some random r_0 and a PCE encryption of m_d. If $\mathcal{A}_g$ queries $(m_d \| \cdot)$ then $\mathcal{B}''_g$ outputs d. (Note that up to this point, the simulation is perfect.) We show that $\mathcal{B}''$ wins the indistinguishability game. If d equals $\mathcal{B}''$'s challenger's bit then $\mathcal{A}_g$ gets two encryptions of the same message; $\mathcal{A}$ is thus playing the unlink game with $b = 1$, for which we assumed $\mathcal{A}_g$ queries the encrypted message to $\mathcal{H}$ with non-negligible probability, in which case $\mathcal{B}''$ wins. On the other hand, if d is different from the challenger's bit (in which case $\mathcal{B}''$ loses) then $\mathcal{A}$ gets encryptions of two different messages and it is thus playing unlink with $b = 0$. For this case however, the previous result for $b = 1$ asserts that $\mathcal{A}$ will not query an encrypted message to the random oracle.

Algorithm $\mathsf{KeyGen}(1^k)$

$(\overline{pk}, \overline{sk}) \xleftarrow{\$} \Pi_d.\mathcal{G}_d(1^k)$
$pk \leftarrow \overline{pk}$
$sk \leftarrow \overline{sk}$
return (pk, sk)

Algorithm $\mathsf{Encrypt}(1^k, pk, m)$

$\overline{pk} \leftarrow pk$
$r \xleftarrow{\$} \{0,1\}^{\ell(k)}$
$\rho \leftarrow \mathcal{H}_1(m\|r)$
$\overline{c_1} \leftarrow \Pi_d.\mathcal{E}_d(1^k, \overline{pk}, \rho)$
$\overline{c_2} \leftarrow m \oplus \mathcal{H}_2(\rho)$
$C \leftarrow (\overline{c_1}, \overline{c_2}, r)$
return C

Algorithm $\mathsf{Decrypt}(1^k, sk, C)$

$(\overline{c_1}, \overline{c_2}, r) \leftarrow C$
$\overline{sk} \leftarrow sk$
$\rho \leftarrow \Pi_d.\mathcal{D}_d(1^k, \overline{sk}, \overline{c_1})$
$m \leftarrow \overline{c_2} \oplus \mathcal{H}_2(\rho)$
if $\rho = \mathcal{H}_1(m||r)$ then return m

Algorithm $\mathsf{PCheck}(1^k, pk, C, m)$

$(\overline{c_1}, \overline{c_2}, r) \leftarrow C$
$\overline{pk} \leftarrow pk$
$\rho \leftarrow \mathcal{H}_1(m\|r)$
$\tilde{c} \leftarrow \Pi_d.\mathcal{E}_d(1^k, \overline{pk}, \rho)$
if $\tilde{c} = \overline{c_1}$ then return 1
else return 0

Fig. 3. Unlinkable PCE from a deterministic encryption scheme Π_d

3.2 A PCE Based on a Deterministic Encryption Scheme

Let $\Pi_d = (\mathcal{G}_d, \mathcal{E}_d, \mathcal{D}_d)$ be a secure deterministic encryption scheme, meaning that it satisfies the ind-det property as defined in [4] and recalled in Sect. 2.2. Let $\mathcal{H}_1 : \{0,1\}^* \to \{0,1\}^{\ell(k)}$ and $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^{\ell(k)}$ be two hash functions modeled as random oracles.

The idea behind this construction is to encrypt with the deterministic encryption algorithm a hash value ρ of the message m together with a random element r and then to compute a one-time pad of the message and the hash value of ρ. We include r in the ciphertext, so knowing m and r, one can recompute the (deterministic) ciphertext and thus perform the plaintext check.

Our random-oracle based construction is detailed in Fig. 3, and Corollary 1 states its security. As we will see, this theorem is a consequence of Theorem 1.

Corollary 1 (sketch, see full version for the full proof). *The PCE construction given in Fig. 3 is unlinkable under the assumption that Π_d is one-way, in the random-oracle model.*

Proof (sketch, see full version for the full proof). This proof is a direct application of Theorem 1 combined with the result from [5] which states that the encryption scheme which consists in computing $c_1 \leftarrow \Pi_d.\mathcal{E}(1^k, pk, r)$ and $c_2 \leftarrow m \oplus \mathcal{H}_2(r)$, where $r \xleftarrow{\$} \{0,1\}^{\ell(k)}$, is ind-cpa if the underlying deterministic encryption scheme Π_d is one-way. □

4 Practical Constructions in the Standard Model

A construction of a secure plaintext-checkable encryption can be proved in the standard model using the technique from [4] for deterministic encryption (see

Fig. 3 of [4]): one replaces the random oracle by a pseudo-random generator [7,24,15] based on a family of trapdoor permutations. As for the previous construction, the idea is to use a secure encryption scheme whose randomness is generated using a secure pseudo-random generator with a seed depending on the message and the random value used to check the plaintext. We here give another practical construction based on the ElGamal encryption scheme [14], which we will then use for our standard-model VLR group signature scheme given in Sect. 5.

4.1 An ElGamal-Based Construction

Our construction lies in an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ where p is a large prime, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map. The elements g and h denote generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. In our scheme, the idea is to encrypt a message m under a public key y using randomness r as $c_1 = my^r, c_2 = g^r$. If we gave $c_3 = h^r$ as well, then using the pairing we can perform plaintext checks since $e(c_1 m^{-1}, g) = e(y, c_3)$. However, this construction does not achieve unlinkability, since we can check whether 2 ciphertexts encrypt the same message by checking whether their quotient encrypts 1. To avoid this, instead of using h as a base for the check element c_3, we use a random base h^a. Since this base is different for every ciphertext, no two ciphertexts can be combined. Our construction is described in Fig. 4 and allows to encrypt messages $m \in \mathbb{G}_1$.

Algorithm $\mathsf{KeyGen}(1^k)$

$x \xleftarrow{\$} \mathbb{Z}_p^*$
$y \leftarrow g^x$
$(pk, sk) \leftarrow (y, x)$
return (pk, sk)

Algorithm $\mathsf{Encrypt}(1^k, pk, m)$

$y \leftarrow pk$
$r, a \xleftarrow{\$} \mathbb{Z}_p^*$
$C \leftarrow (my^r, g^r, h^a, h^{ar})$
return C

Algorithm $\mathsf{Decrypt}(1^k, sk, C)$

$x \leftarrow sk$
$(c_1, c_2, c_3, c_4) \leftarrow C$
if $e(g, c_4) \neq e(c_2, c_3)$ then return $\perp$
$m \leftarrow c_1 / c_2^x$
return m

Algorithm $\mathsf{PCheck}(1^k, pk, C, m)$

$y \leftarrow pk$
$(c_1, c_2, c_3, c_4) \leftarrow C$
if $e(g, c_4) \neq e(c_2, c_3)$ then return 0
if $e(c_1/m, c_3) = e(y, c_4)$ then return 1
else return 0

Fig. 4. Unlinkable PCE in the standard model

4.2 Security Arguments

To prove unlinkability of the construction in Fig. 4, we introduce a new assumption (whose security in the generic-group model is proved in the full version), which combines features of the Decision Linear Assumption (DLIN) and the assumption that DDH holds in both base groups of an asymmetric bilinear group (known as "SXDH").

Assumption 1 *Given an asymmetric bilinear group* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *with generators* $g \in \mathbb{G}_1$ *and* $h \in \mathbb{G}_2$*, and the tuple* $(g^x, g^{rx}, g^{sx}, h^a, h^{ar}, h^b, h^{br}, V)$ *for random* $x, r, s, a, b \in \mathbb{Z}_p$*, it is hard to decide if* $V = g^{r+s}$ *or* V *is random in* $\mathbb{G}_1$*.*

Let us first analyze the $\mathbb{G}_1$ part of our assumption: (g^x, g^{rx}, g^{sx}) and g^{r+s}. DLIN states that given $(g^x, g^y, g^{rx}, g^{sy})$ it is hard to distinguish g^{r+s} from random. The $\mathbb{G}_1$ components of our assumption can thus be seen as a DLIN instance with $y = x$ (note that whereas DLIN also holds in *symmetric* groups, this is not the case when $y = x$). It is also immediate that this "partial" assumption is a DDH instance where $s = 0$, and thus implied by DDH. However, since—as opposed to DDH—we have *two* random *combined* exponents r and s for the challenge, this allows us to add values depending on them in $\mathbb{G}_2$, which cannot be used to verify the structure of g^{r+s}, since the bases h^a and h^b for r and s are different.

The following theorem holds against adversaries $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ where $\mathcal{A}_f$ outputs the uniform distribution. This restriction is similar to the results by Bellare et al. [3] for their practical construction of a deterministic encryption scheme. In fact, in real life applications, the uniform distribution is most of time enough and easily obtained. In particular, this notion also suffices when applying the scheme to VLR group signatures.

Theorem 2. *Under Assumption 1, the construction from Fig. 4 is a PCE scheme which is* unlink *against adversaries outputting the uniform distribution.*

5 Application to VLR Group Signature

In this section we use our new primitive as a building block for group signatures with verifier-local revocation (VLR) [10]. This is a group signature scheme [2,8] which allows an efficient revocation of group members.

Our aim in this section is twofold. First, we present plaintext-checkable encryption as a new building block for group signatures with VLR; thus any improvement to PCE is likely to lead to more efficient group signatures with VLR. Second, we design in the following, to the best of our knowledge, the most efficient group signature scheme with VLR and backward unlinkability in the standard model. We first recall the concept of group signatures with VLR, and eventually describe our new construction.

5.1 Definitions for Group Signatures with VLR

Let k, n and T be integers. A *group signature scheme with VLR* (VLR-GS for short) is composed of the following algorithms (following [19]).

– KeyGen takes as input a security parameter 1^k, the number n of group members and the number T of time periods. It produces the group public key gpk, an n-element vector of user keys $\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and an $(n \times T)$-element vector of user revocation tokens $\mathbf{grt} = (\mathsf{grt}[1][1], \ldots, \mathsf{grt}[n][T])$.

– Sign takes as input the group public key gpk, the current time interval j, a secret key sk_i for $i \in [\![1, n]\!]$ of a group member and a message $m \in \{0,1\}^*$, and outputs a signature σ.
– Verify takes as input the group public key gpk, the current time period j, the public key of the revocation authority rpk, a set of revocation tokens RL_j, and a purported signature σ on a message m. It returns either valid if the signature σ is valid or invalid if σ is not a valid signature or if the user who generated it has been revoked.

The security requirements are *traceability* and *backward unlinkability (BU) anonymity*. The corresponding formal definitions can be found in [19]. We only recall the BU-anonymity since adding the VLR functionality to a group signature scheme only concerns this security notion, whereas traceability is inherited from the original scheme. A VLR-GS with backward unlinkability is BU-anonymous if no p.p.t. adversary $\mathcal{A}$ has non-negligible advantage in the following game.

1. The challenger $\mathcal{C}$ executes $(\mathsf{gpk}, \mathbf{sk}, \mathbf{grt}) \xleftarrow{\$} \mathsf{KeyGen}(1^k, n, T)$ and the adversary is given gpk.
2. For each period, $\mathcal{C}$ increments the counter j and during this period, $\mathcal{A}$ can access the $\mathsf{Sign}(\cdot,\cdot)$ oracle, which gives a group signature on a message m by a user i during time period j, the $\mathsf{Corrupt}(\cdot)$ oracle, which permits to corrupt the user i and the $\mathsf{Revoke}(\cdot)$ oracle, which revokes the member i.
3. At some period $j^* \in [1, T]$, $\mathcal{A}$ outputs (m^*, i_0, i_1) such that i_0 and i_1 are not corrupted and have not been revoked during or before the time period j^*. The challenger $\mathcal{C}$ flips a coin b and generates $\sigma^* \xleftarrow{\$} \mathsf{Sign}(\mathsf{gpk}, j^*, \mathsf{sk}_{i_b}, m^*)$, which is sent to $\mathcal{A}$.
4. $\mathcal{A}$ can again access the above oracles. $\mathcal{A}$ is not allowed to corrupt i_0 nor i_1 but it may revoke them after time period j^*.
5. Eventually, $\mathcal{A}$ outputs a bit b^* and wins if $b = b^*$.

The advantage of $\mathcal{A}$ in breaking this anonymity is defined as $\mathbf{Adv}^{\mathsf{bu\text{-}a}}_{\mathsf{VLR\text{-}GS},\mathcal{A}}(k) := |\Pr[b = b^*] - \frac{1}{2}|$.

5.2 Using PCE for Group Signatures with VLR

Starting with a Group Signature Scheme. For concreteness, we base our instantiation on the group signature scheme by Fuchsbauer and Abe et al. in [12,1], which is itself based on Groth's scheme [16], which makes use of the non-interactive zero-knowledge (NIZK) proofs from Groth and Sahai [17].

In a nutshell, each user creates a key pair for an *automorphic signature* scheme[3] [12,1]. The group public key is a signature verification key, whose corresponding signing key is used by the group manager to sign a user's verification

[3] A signature scheme defined over a bilinear group is *automorphic* if the verification keys lie in the message space, and if the messages and the signatures consist of group elements. The first property enables certification of keys, whereas the second makes it possible to give efficient NIZK proofs of knowledge of valid signatures and messages using Groth-Sahai proofs.

key when he joins the group. To make a group signature, the user first signs the message using his personal signing key; the group signature is then a Groth-Sahai proof of knowledge of the following: the user's verification key, a valid certificate on it by the group manager, and a signature on the message that is valid under his verification key. Since the registration protocol consists of only one round, the scheme is concurrently secure. Moreover, since the group members create their own signing keys, the scheme achieves *non-frameability* [6].

Adding the VLR Property. When adding verifier-local revocability, to achieve backward unlinkability, we use the system due to Nakanishi and Funabiki [20]. This consists in defining time periods and constructing one key (called the revocation token) per group member and time period. This token is to be used by the group member when making a group signature. When a member is revoked, all the revocation tokens related to the revoked group member and future time periods are published. These public revocation tokens are then used by the verifier to check whether the received group signature has been produced with a published value, and thus by a revoked group member.

Making Use of a PCE. The group signature cannot contain the revocation token in the clear, as this would compromise the member's anonymity. Our approach is to include in the group signature a *plaintext-checkable encryption* of the revocation token, together with a proof of well-formedness. When a revoked group member's token gets published, the verifier can use PCheck of the PCE scheme to check whether the group signature comes from a revoked member or not. For our concrete scheme, we use the standard-model PCE scheme from Sect. 4.1, since it complies with the Groth-Sahai methodology.

5.3 Our Concrete Instantiation

We will use the group signature scheme on which we base our construction as a black box and simply add one PCE encryption and a proof of consistency to make it a VLR scheme. We require that the group signature is a Groth-Sahai proof of knowledge in an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ and that the user verification key contains a component h^{v_i}, where v_i is the i-th user's signing key. (This is the case e.g. in the construction from [12,1]).

In the setup phase of scheme (when the common reference string for Groth-Sahai proofs is created), we now also create a key pair $(y = g^x, x) \in \mathbb{G}_1 \times \mathbb{Z}_p$ for our PCE scheme from Sect. 4.1 and add y to the public parameters. As in [20,19], we introduce a vector $(P_1, \ldots, P_T)$ of $\mathbb{G}_1$ elements, where T is the maximum number of time periods. The revocation token for user i (holding secret key v_i) for time interval j is defined as $P_j^{v_i}$.

When creating a group signature, the user must additionally encrypt his token for the current time interval and prove that it is well-formed. The token is of the form P^v, so we need to prove that v is the same as in the user verification key element $w := h^v$ (of which the group signature will prove knowledge). The PCE encryption of the token is $C = (C_1, C_2, C_3, C_4) = (P^v y^r, g^r, h^a, h^{ar})$. To

prove well-formedness, we introduce an auxiliary variable $z := h^r$, of which we also prove knowledge in the group signature. Groth-Sahai proofs allow us to prove knowledge of group elements that satisfy *pairing-product equations* (PPE). Let v be such that P^v is the plaintext of C. Then the following PPEs assert that $w = h^v$ (the group elements of which we prove knowledge are underlined): $e(C_1, h) = e(P, \underline{w})\, e(y, \underline{z})$ and $e(C_2, h) = e(g, \underline{z})$.

In addition to C, we include in the group signature a Groth-Sahai NIZK proof that the above equations are satisfied. Our new verification procedure now additionally checks this new proof component, and runs PCheck on C and the elements of the revocation list to check if the user has been revoked.

We note that our techniques also work if the verification key contains g^v rather than h^v: we can introduce a second encrypted auxiliary variable $z' := h^v$ and add a proof of $e(\underline{g^v}, h) = e(g, \underline{z'})$. We have thus shown that adding to a Groth-Sahai based group signature scheme (with user verification keys containing a generator to the power of the signing key) a plaintext-checkable encryption of a token, gives a group signature scheme with VLR and backward unlinkability.

5.4 Backward-Unlinkable Anonymity

We outline the proof that our scheme satisfies backward-unlinkable anonymity. The proof proceeds by a series of games. The first game is the experiment defined in Sect. 5.1. In the second game, instead of running KeyGen, we compute the common reference string for Groth-Sahai proofs in a way that will lead to perfectly hiding proofs of knowledge, which can be simulated. By the zero-knowledge property of Groth-Sahai proofs, the first two games are indistinguishable. In Game 3, the challenger picks 2 random users, hoping they will be the challenge users i_0 and i_1 output by the adversary in Step 3 of the game. If the challenger did not guess these users correctly, it aborts the game. This introduces a polynomial loss in the security reduction.

In Game 4 the challenger simulates the NIZK proofs in the following signatures it gives to the adversary: all signatures in signing queries for users i_0 and i_1 queried up to the challenge time period j^*; and the challenge signature σ^*. It follows from the zero-knowledge property of Groth-Sahai proofs that Game 4 is indistinguishable from Game 3.

We can now play with the plaintext-checkable encryptions C of tokens which are given to the adversary as part of the simulated group signature (either in a signing query for users i_0 and i_1 in time $j < j^*$ or the challenge signature). Since the proof of consistency of these C's is simulated, we can change the actual values, which we will do in the following. Next, when computing the values P_j during setup, the challenger sets them as $P_j := g^{d_j}$ and stores d_j. We now define a series of games, in which, one by one, we replace tokens $P_1^{v_{i_0}}, \ldots, P_{j^*}^{v_{i_0}}$ and tokens $P_1^{v_{i_1}}, \ldots, P_{j^*}^{v_{i_1}}$ by random values. This is reduced to the DDH assumption, which implies that given values g^d and g^v, we can replace g^{dv} by a random value. Note that given a DDH challenge, the challenger can use the logarithms d_j to compute the values $P_j^{v_i}$ it is not changing in that step.

After this series of games, the only dependency of the challenge signature on the bit b occurs when the adversary asks for a signature of user i_b in time interval j^*. Since the tokens are chosen uniformly at random, we can replace the encryption of the token in the challenge signature by a random value. This is implied by unlinkability of our PCE scheme (which states that two encryptions of the same value are indistinguishable from two encryptions of two different (random) values). After this final step the challenge signature is independent of b and the adversary's winning probability is thus exactly $\frac{1}{2}$.

5.5 Comparison with Related Work

Regarding related work on group signature schemes with VLR, there are typically 3 criteria to compare such schemes: random-oracle or standard model, anonymity revocation or not and backward unlinkability or not. Table 1 compares all existing solutions, to the best of our knowledge.

Table 1. Related work on group signatures with VLR

Papers	Standard model	Anonymity revocation	Backward unlinkability
[10]	No	No	No
[20,21,25]	No	Yes	Yes
[19]	Yes	(Yes)	Yes
Ours	Yes	(Yes)[4]	Yes

Achieving CCA Security. An additional property not considered in the above table is CCA-anonymity, meaning the scheme remains anonymous even if the adversary has an oracle to open signatures of its choice, as considered e.g. in the model by Bellare et al. [6]. This notion is achieved by variants of the group signature schemes on which we base our VLR scheme, using one-time signatures and a weakly CCA tag-based encryption scheme, as proposed by Groth in [16].

The tag-based encryption scheme used is Kiltz's construction [18] is secure under the DLIN assumption [8] and is defined over symmetric bilinear groups. As DDH is easy in such groups, our PCE scheme would not be secure and can thus not be added to these schemes. We believe however that starting from *linear encryption* [8] rather than ElGamal, and adding elements enabling plaintext checkability, one could define a PCE scheme over *symmetric* bilinear groups.

Efficiency Considerations. We can now compare the efficiency of standard model group signatures with VLR and backward unlinkability, which amounts to comparing us with the scheme by Libert and Vergnaud [19]. On one hand, regarding [19], a group signature is composed of 46 elements in $\mathbb{G}$ and 1 element in $\mathbb{G}_T$. The time complexity of a group-signature creation necessitates 2 modular exponentiations in $\mathbb{G}$, 6 commitment generations, 2 quadratic GS proofs and 4 linear GS proofs. The revocation checking requires the computation of

[4] Not explicitly detailed but can be easily added by giving the trapdoor for the CRS of Groth-Sahai proofs to the opener.

one pairing per element in RL_j. On the other hand, our signatures are composed of 12 elements in $\mathbb{G}_1$, 18 elements in $\mathbb{G}_2$ and no element in $\mathbb{G}_T$. The signer must perform 6 modular exponentiations, 1 quadratic GS proofs and 5 linear GS proofs. The revocation checking requires the computation of 2 pairings per element in RL_j. Considering moreover that in asymmetric groups, representations of group elements are shorter and computation of pairings are much more efficient, our scheme is more efficient in terms of signature computation and size but necessitates slightly more work during the revocation check.

6 Conclusion

We proposed a new promising public-key encryption scheme with a special feature: this primitive allows anyone to verify whether a given ciphertext (together with the public key used to encrypt) actually encrypts any potential message. However, if the messages come from a space with enough entropy, one cannot decide whether two ciphertexts encrypt the same message. Plaintext-checkable encryption with unlinkable ciphertexts is perfectly adapted to design group signatures with verifier-local revocation and backward unlinkability. The efficiency of the constructions also enables its use in a context of cloud storage services.

Acknowledgements. This work has been supported by the French Agence Nationale de la Recherche under the PACE 07 TCOM Project, the European Commission under Contract ICT-2007-216676 ECRYPT II and EPSRC Grant EP/H043454/1. We are grateful to Jacques Traoré for his suggestions of improvement, and to the anonymous referees for their valuable comments.

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
2. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and Efficiently Searchable Encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
4. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)
5. 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 (1993)
6. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
7. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput. 13(4), 850–864 (1984)

8. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
9. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
10. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security, pp. 168–177. ACM (2004)
11. Camenisch, J., Groth, J.: Group Signatures: Better Efficiency and New Theoretical Aspects. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 120–133. Springer, Heidelberg (2005)
12. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), http://eprint.iacr.org/
13. Fuhr, T., Paillier, P.: Decryptable Searchable Encryption. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 228–236. Springer, Heidelberg (2007)
14. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
15. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Proc. of STOC 1989, pp. 25–32. ACM (1989)
16. Groth, J.: Fully Anonymous Group Signatures without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
17. Groth, J., Sahai, A.: Efficient Non-Interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
18. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
19. Libert, B., Vergnaud, D.: Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 498–517. Springer, Heidelberg (2009)
20. Nakanishi, T., Funabiki, N.: Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005)
21. Nakanishi, T., Funabiki, N.: A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 17–32. Springer, Heidelberg (2006)
22. Ostrovsky, R., Skeith III, W.E.: Private searching on streaming data. J. Cryptology 20(4), 397–430 (2007)
23. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic Public Key Encryption with Equality Test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010)
24. Yao, A.C.-C.: Theory and applications of trapdoor functions (extended abstract). In: Proc. of FOCS 1982, pp. 80–91. IEEE (1982)
25. Zhou, S., Lin, D.: Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 126–143. Springer, Heidelberg (2006)

Generic Construction of Chosen Ciphertext Secure Proxy Re-Encryption

Goichiro Hanaoka[1], Yutaka Kawai[2], Noboru Kunihiro[2], Takahiro Matsuda[1], Jian Weng[3], Rui Zhang[4], and Yunlei Zhao[5]

[1] National Institute of Advance Industrial Science and Technology (AIST)
{hanaoka-goichiro,t-matsuda}@aist.go.jp
[2] The University of Tokyo
kawai@it.k.u-tokyo.ac.jp, kunihiro@k.u-tokyo.ac.jp
[3] Department of Computer Science, Jinan University
cryptjweng@gmail.com
[4] SKLOIS, Institute of Software, Chinese Academy of Sciences
r-zhang@is.iscas.ac.cn
[5] Software School, Fudan University
ylzhao@fudan.edu.cn

Abstract. In this paper, we present the first generic construction of a chosen-ciphertext (CCA) secure uni-directional proxy re-encryption (PRE) scheme. In particular, full CCA security (i.e., not relaxed CCA security such as replayable CCA security) of our proposed scheme is proven even against powerful adversaries that are given a more advantageous attack environment than in all previous works, and furthermore, random oracles are not required. To achieve such strong security, we establish a totally novel methodology for designing PRE based on a specific class of threshold encryption. Via our generic construction, we present the first construction that is CCA secure in the standard model.

1 Introduction

Proxy re-encryption (PRE) is an interesting extension to traditional public key encryption (PKE). In addition to the normal operations of PKE, with a dedicated re-encryption key (generated by receiver A), a proxy can turn a class of ciphertexts destined for user A into those for user B. A remarkable property of PRE is that the proxy carrying out the transform is totally ignorant of the plaintext. PRE was first formalized by Blaze et al. [5] and has received much attention in recent years. There are many models as well as implementations; refer to [5,3,9,13,17,10,18,19] for some examples. However, as pointed out in [21], the design of a chosen-ciphertext (CCA) secure uni-directional PRE scheme without random oracles remains unsolved. In this paper, we present the *first* uni-directional PRE scheme with CCA security in the standard model, in the sense of [13], thus solving the above open problem. Moreover, our scheme achieves the strongest security to date.

At first glance, it appears to be easy to extend the replayable-CCA (RCCA) secure scheme to full CCA security in the same model [13], but this turns out to

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 349–364, 2012.

be an extremely challenging problem. The reason is that in a PRE system, proxy re-encryption uses a re-encryption key to transform a ciphertext for Receiver A to one for another receiver B (and at this point, there is already some sort of malleability within). However, CCA security explicitly requires that no meaningful modifications can be made to any ciphertext, and therefore it is quite difficult to achieve both requirements simultaneously. We note that in the RCCA model, a challenger ignores all decryption queries decrypted to the challenge plaintext pair; in other words, RCCA security permits malleability of the challenge plaintext. As a result, it is quite difficult to upgrade RCCA secure PRE schemes to CCA secure ones.

Main Difficulty. We identify two problems in a CCA secure construction of PRE. The first stems from the contradictive requirements of the re-encryption functionality of PRE and the tamper-proof property by CCA security. Usually, good mathematical structures are demanded when efficiently transforming a ciphertext for receiver A into another for receiver B. Consider a discrete-log type PRE scheme, say the ElGamal-based scheme put forth by Blaze et al. [5]. It is easily seen that the scheme is not CCA secure, because a ciphertext c with plaintext m can be modified to another ciphertext c' with plaintext m. Querying the decryption oracle with c' enables the adversary to recover m.

The second difficulty comes from the fact that the proxy has no idea of the decryption key, and it may not be able to determine whether a ciphertext is valid. However, a simulator, which has to answer all re-encryption queries correctly, may leak useful information to the adversary, thereby eventually causing the simulation to fail.

Our Contributions. First, we give a CCA security definition for PRE, which naturally extends the RCCA one given in [13]. While our definition is the strongest to date, it is indeed quite natural, because we give an adversary all the possible resources, except those that allow it to trivially win the game.

Second, we propose a new methodology for building secure PRE schemes in our CCA-security model. There are three ingredients in our generic construction: resplittable TPKE, PKE, and digital signature. Though resplittable TPKE sounds like a new primitive, we show that it is not a luxury one at all: many known TPKE schemes already satisfy the requirements.

To summarize, our approach is based on the following. We observe that a primitive TPKE [11,20,8,6], faces similar problems to those mentioned above. In a TPKE scheme, the decryption power is split among n decryption servers. Each decryption server holds only a part of the secret key and upon receipt of a ciphertext carries out partial decryption and outputs a plaintext share. Combining any set containing at least t different shares (t is called the threshold), one can recover the plaintext; fewer than t shares does not provide sufficient information to reconstruct it. To achieve threshold decryption, a TPKE scheme usually has good mathematical structures; in fact, well-known TPKE schemes based on the discrete-log problem [20,8,6] all utilize such good mathematical structures. Secondly, since each decryption server does not hold the decryption

key, it may not be able to distinguish valid/invalid ciphertexts. Nevertheless, it still has to answer partial decryption queries in the CCA sense.

Owing to their possible connections with respect to the above, we then considered that some of the design strategies of CCA-secure TPKE could be useful in constructing CCA-secure PRE.

Related Works. Mambo and Okamoto introduced the concept of proxy decryption [14]. Later, Ivan and Dodis [12] proposed a generic construction of proxy cryptography based on sequential multiple encryption. Neither of these works considered the re-encryption functionality.

Blaze, Bleumer and Strauss formulated the concept of PRE cryptosystems [5] and proposed the first bidirectional PRE scheme based on ElGamal. Subsequently, Ateniese et al. [3], Canetti and Hohenberger [9], Libert and Vergnaud [13], Chow et al. [10], and Shao et al. [18,19] proposed different PRE schemes with various properties.

Shao and Cao [17] proposed a PRE scheme without pairings. Later, however, Zhang et al. pointed out that it is not secure in the Libert-Vergnaud security model [22]; that is, it does not provide master key security. Subsequently, Matsuda et al. proposed a PRE scheme without pairings [15], but recently Weng, Zhao and Hanaoka [21] pointed out that their scheme is not chosen-ciphertext secure. Thus prior to this work, the construction of CCA secure PRE has remained unsolved.

2 Preliminaries

2.1 Public Key Encryption

Syntax. A public key encryption scheme (PKE) consists of three algorithms $(\mathtt{PKG}, \mathtt{PEnc}, \mathtt{PDec})$.

$\mathtt{PKG}$ takes a security parameter Λ as input, and outputs a decryption key dk and a public key pk, denoted as $(dk, pk) \leftarrow \mathtt{PKG}(\Lambda)$.

$\mathtt{PEnc}$ takes a public key pk and a plaintext m in a plaintext space defined by pk as input, and outputs a ciphertext ψ, denoted as $\psi \leftarrow \mathtt{PEnc}(pk, m)$.

$\mathtt{PDec}$ takes a decryption key dk and a ciphertext ψ as input, and outputs a decryption result m (or a special symbol $\perp$ meaning the ciphertext is invalid), denoted as $m \leftarrow \mathtt{PDec}(dk, \psi)$.

We require the standard correctness for a PKE scheme, namely, for any $(dk, pk) \leftarrow \mathtt{PKG}(\Lambda)$ and any plaintext m, we have $m = \mathtt{PDec}(dk, \mathtt{PEnc}(pk, m))$.

Chosen Ciphertext Security in the Multi-User Setting [4]. The widely accepted security definition of PKE is indistinguishability against chosen ciphertext attack (IND-CCA). We will use its multi-user version [4] which is polynomially equivalent to the ordinary IND-CCA security. The multi-user IND-CCA security is defined by the following game between a challenger and an adversary A. First, the challenger picks the challenge bit $b \in \{0, 1\}$, computes

$(dk_i, pk_i) \leftarrow \mathtt{PKG}(\Lambda)$ for $i \in \{1, \ldots, n\}$, and gives $\{pk_i\}_{i=1}^n$ to A. A can adaptively make decryption and LR queries. For a LR query (i, m_0, m_1) where i is an index and (m_0, m_1) is a message pair, the challenger computes $c \leftarrow \mathtt{PEnc}(pk_i, m_b)$, and then returns c to A. For a decryption query (i, c) where i is an index and c is a ciphertext, the challenger computes $m \leftarrow \mathtt{PDec}(dk_i, c)$, and returns m to A, except that if A has previously asked a LR query (i, m_0, m_1) and c was its answer, then the challenger returns $\perp$ to A. Finally, A outputs a guess bit b' for b. A wins the game if $b = b'$. We say a PKE scheme is IND-CCA secure, if any probabilistic polynomial adversary A's advantage $\mathrm{Adv}^{\mathrm{CCA-PKE}}_{(\mathsf{A},n)}(\Lambda) = |\Pr[b = b'] - \frac{1}{2}|$ in the above game is negligible. When $n = 1$, this security is the ordinary IND-CCA security [16]. In this case, we describe the advantage as $\mathrm{Adv}^{\mathrm{CCA-PKE}}_{\mathsf{A}}(\Lambda)$, simply.

2.2 Strongly Unforgeable Signature

Syntax. A digital signature scheme consists of three algorithms $(\mathtt{SKG}, \mathtt{Sign}, \mathtt{SVer})$.

`SKG` takes a security parameter Λ as input, and outputs a signing key sk and a verification key vk, denoted as $(sk, vk) \leftarrow \mathtt{SKG}(\Lambda)$.

`Sign` takes a signing key sk and a message m as input, and outputs a signature σ, denoted as $\sigma \leftarrow \mathtt{Sign}(sk, m)$.

`SVer` takes a verification key vk, a message m, and a signature σ as input, and outputs valid (meaning that the signature σ is a valid signature on m under vk) or $\mathsf{invalid}$.

We require the standard correctness for a signature scheme, namely, for any $(sk, vk) \leftarrow \mathtt{SKG}(\Lambda)$ and any message m, we have $\mathsf{valid} = \mathtt{SVer}(vk, m, \mathtt{Sign}(sk, m))$.

Strong Unforseability [1]. The widely accepted security definition of digital signature is strong unforgeability, which is defined by the following game between a challenger and an adversary A. In the strong unforgeability game, A is given vk where $(sk, vk) \leftarrow \mathtt{SKG}(\Lambda)$, and can make signing queries. For the i-th signing query on a message m_i, the challenger computes $\sigma_i \leftarrow \mathtt{Sign}(sk, m_i)$, returns σ_i to A, and stores (m_i, σ_i). Finally, A outputs a forgery (m^*, σ^*). A wins the game if $\mathsf{valid} \leftarrow \mathtt{SVer}(vk, m^*, \sigma^*)$ and $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for any i. We say a digital signature scheme is strongly unforgeable, if any probabilistic polynomial adversary A's advantage $\mathrm{Adv}^{\mathrm{SUF}}_{\mathsf{A}}(\Lambda) = \Pr[\mathsf{A}\text{ wins}]$ in the above game is negligible. If the advantage is negligible for an adversary that makes only one signing query in the above game, a signature scheme is called a one time signature.

3 Single Use Unidirectional Proxy Re-Encryption

In this section, we present the model and the security definition. We focus on single-use unidirectional proxy re-encryption (SUPRE) schemes. First, we review the syntax of SUPRE scheme. Second, we define first/second level CCA security which is stronger than previous RCCA security [13]. Finally, we explain the difference between our definitions and previous definitions.

Syntax. A single-use unidirectional proxy re-encryption scheme consists of the following six algorithms ($\mathtt{KG}$, $\mathtt{RKG}$, $\mathtt{Enc}$, $\mathtt{REnc}$, $\mathtt{Dec}_1$, $\mathtt{Dec}_2$):

$\mathtt{KG}$ takes as input the security parameter Λ and generates a secret key sk and a public key pk, denoted as $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KG}(\Lambda)$.

$\mathtt{RKG}$ takes as input a secret key sk_i of user i and a public key pk_j of user j, and outputs a unidirectional re-encryption key $rk_{i\to j}$, denoted as $rk_{i\to j} \leftarrow \mathtt{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$.

$\mathtt{Enc}$ takes as input a public key pk_i of user i and a plaintext m, and outputs a *second level* ciphertext c that can be re-encrypted for another party.

$\mathtt{REnc}$ takes as input a second level ciphertext c_i to user i and a re-encryption key $rk_{i\to j}$ and outputs a *first level* ciphertext $\widehat{c}_j$ for user j or a special symbol $\perp$ if c_i is invalid, denoted as $\widehat{c}_j \leftarrow \mathtt{REnc}(rk_{i\to j}, c_i)$.

$\mathtt{Dec}_1$ takes as input a secret key sk_i of user i and a first level ciphertext $\widehat{c}_i$ as input, and outputs a plaintext m or $\perp$ (indicating "$\mathtt{invalid\ ciphertext}$"), denoted as $m \leftarrow \mathtt{Dec}_1(\mathsf{sk}_i, \widehat{c}_i)$.

$\mathtt{Dec}_2$ takes as input a secret key sk_i of user i and a second level ciphertext c_i and outputs a plaintext m or $\perp$.

We require the correctnesses for a SUPRE scheme as follows: (1)For any plaintext m and for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KG}(\Lambda)$, we have $m = \mathtt{Dec}_2(\mathsf{sk}, \mathtt{Enc}(\mathsf{pk}, m))$. (2) For any plaintext m and for any $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathtt{KG}(\Lambda)$ and $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \mathtt{KG}(\Lambda)$, we have $m = \mathtt{Dec}_1(\mathsf{sk}_j, \mathtt{REnc}(\mathtt{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), \mathtt{Enc}(\mathsf{pk}_i, m)))$.

Security Definition. Here, we give the formal definitions of CCA security of SUPRE. We first describe our formal security definitions, and then explain their features in detailed in Sec. 2.1.

First, we define the security for second level ciphertext. We define the security of a SUPRE scheme using the following game between a challenger and an adversary A.

Setup. The challenger generates honest users' key pairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathtt{KG}(\Lambda)$ for $i = 1$ to n and sets $\mathcal{PK} = \{\mathsf{pk}_i\}_{i=1}^{n}$. Next, the challenger generates a challenge user's key pair $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathtt{KG}(\Lambda)$. Then, the challenger gives the security parameter Λ and $\mathcal{PK}^* = \mathcal{PK} \cup \{\mathsf{pk}_{i^*}\}$ to A.

Re-Encryption Key Generation Query. For a re-encryption key generation query $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j)$, where pk_j is an arbitrary public key chosen by A, the challenger responds as follows. If $\mathsf{pk}_i = \mathsf{pk}_{i^*}$ and $\mathsf{pk}_j \notin \mathcal{PK}^*$, then the challenger returns the special symbol $\perp$ to A Otherwise, the challenger responds with $\mathtt{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$.

Re-Encryption Query. For a re-encryption query $(\mathsf{pk}_i \in \mathcal{PK}^*, \mathsf{pk}_j, c_i)$, where pk_j is an arbitrary public key chosen by A, the challenger responds as follows. If $(\mathsf{pk}_i, c_i) = (\mathsf{pk}_{i^*}, c_{i^*})$ and $\mathsf{pk}_j \notin \mathcal{PK}^*$, then the challenger

returns the special symbol $\perp$ to A. Otherwise, the challenger responds with $\mathtt{REnc}(\mathtt{RKG}(\mathsf{sk}_i, \mathsf{pk}_j), c_i)$.[1]

Challenge Query. This query is asked only once. For a challenge query (m_0, m_1), the challenger picks a random $b \in \{0,1\}$ and computes $c_{i^*} \leftarrow \mathtt{Enc}(\mathsf{pk}_{i^*}, m_b)$. Then it gives c_{i^*} to A.

First Level Decryption Query. For a first level decryption query $(\mathsf{pk}_i \in \mathcal{PK}^*, \widehat{c}_i)$, the challenger responds as follows: If A has asked a re-encryption query $(\mathsf{pk}_{i^*}, \mathsf{pk}_i \in \mathcal{PK}, c_{i^*})$ and obtained $\widehat{c}_i$ previously, then the challenger returns $\perp$ to A. Else if A has asked a re-encryption key query $(\mathsf{pk}_{i^*}, \mathsf{pk}_i \in \mathcal{PK})$ previously and $\mathtt{Dec}_1(\mathsf{sk}_i, \widehat{c}_i) \in \{m_0, m_1\}$, then the challenger returns the special symbol $\mathtt{test}$ to A. Otherwise, the challenger responds with $\mathtt{Dec}_1(\mathsf{sk}_i, \widehat{c}_i)$.

Second Level Decryption Query. For a second level decryption query $(\mathsf{pk}_i \in \mathcal{PK}^*, c_i)$, the challenger responds with $\mathtt{Dec}_2(\mathsf{sk}_i, c_i)$, except that if $(\mathsf{pk}_i, c) = (\mathsf{pk}_{i^*}, c_{i^*})$, then the challenger returns a special symbol $\perp$.

Finally, A outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$. We define the advantage of A as $\mathrm{Adv}^{second}_{(\mathsf{A},n)}(\Lambda) = |\Pr[b = b'] - \frac{1}{2}|$.

Definition 1 (Second Level CCA-SUPRE Security). *We say a SUPRE scheme is second level CCA-SUPRE secure, if for any probabilistic polynomial adversary* A *and for all positive polynomials* n*, the advantage* $Adv^{second}_{(\mathsf{A},n)}(\Lambda)$ *is negligible.*

Next, we define the security for first level ciphertexts with the following game between an adversary A and a challenger.

Setup. The challenger generates a challenge public/secret keys, $(\mathsf{sk}_{i^*}, \mathsf{pk}_{i^*}) \leftarrow \mathtt{KG}(\Lambda)$. The challenger gives security parameter Λ and pk_{i^*} to A.

Re-Encryption Key Generation Query. For a re-encryption key generation query pk, where pk is a public key of A's choice (for which A is not required to reveal the secret key), the challenger responds with $\mathtt{RKG}(\mathsf{sk}_{i^*}, \mathsf{pk})$.

Challenge Query. This query is asked only once. For a challenge query $(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A}, m_0, m_1)$ where $(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$ is required to be a valid key pair, the challenger picks the challenge bit $b \in \{0,1\}$ randomly and computes $c \leftarrow \mathtt{Enc}(\mathsf{pk}_\mathsf{A}, m_b)$ and $\widehat{c}_{i^*} \leftarrow \mathtt{REnc}(\mathtt{RKG}(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_{i^*}), c)$. It then returns $\widehat{c}_{i^*}$ to A.

First Level Decryption Query. For a first level decryption query $\widehat{c}$, the challenger responds with $\mathtt{Dec}_1(\mathsf{sk}_{i^*}, \widehat{c})$, except that if $\widehat{c} = \widehat{c}_{i^*}$, then the challenger returns a special symbol $\perp$ to A.

Second Level Decryption Query. For a second level decryption query c, the challenger responds with $\mathtt{Dec}_2(\mathsf{sk}_{i^*}, c)$.

Finally, A outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$. We define the advantage of A as $\mathrm{Adv}^{first}_{\mathsf{A}}(\Lambda) = |\Pr[b = b'] - \frac{1}{2}|$.

[1] Note that in the security model defined above a fresh re-encryption key will be used in each re-encryption query. We can further strengthen this security model by allowing the adversary to ask re-encryption queries under previously used re-encryption keys. We will discuss the detail in the full version of this paper.

Definition 2 (First Level CCA-SUPRE Security). *We say a SUPRE scheme is first level CCA-SUPRE secure, if for any probabilistic polynomial adversary* A*, the advantage* $Adv_{\mathsf{A}}^{first}(\Lambda)$ *is negligibleD*

Definition 3 (CCA-SUPRE Security). *We say that a SUPRE scheme is CCA-SUPRE secure if the scheme is first level CCA-SUPRE secure and second level CCA-SUPRE secure.*

3.1 Difference from Previous Security Definitions

CCA Security. Note that as in the RCCA security definitions, a situation still occurs in our second level security model whereby the adversary receives a special symbol "test" as an answer to a first level decryption query, which means that the decryption result of the (first level) ciphertext submitted to the first level decryption oracle is one of the challenge plaintexts (submitted as the second level challenge). As such, one might question whether this security model is stronger than previous RCCA security models. The reason that our second level security model still considers the special symbol "test" is because there is a trivial CCA attack against an adversary who (1) first asks a re-encryption key generation query from the challenge key pk_{i^*} to a honest user's key pk_j and obtains $rk_{i^* \to j}$, (2) re-encrypts the challenge ciphertext c^* into a first level ciphertext $\widehat{c}$ using it, and (3) asks a first level decryption query of $\widehat{c}$ with the honest user's pk_j. We find it impossible to avoid this trivial attack, *as long as we stick to the current syntax for PRE schemes*.

Note that if an adversary does not ask such a problematic re-encryption key generation query, then from the adversary's viewpoint our first level decryption oracle works as an ordinary decryption oracle, and in particular, even if the decryption result of some first level ciphertext is one of the challenge plaintexts, our first level decryption oracle returns a correct decryption result to the adversary. Hence, our CCA security definition is stronger than that of RCCA security in [13].

Simplification of Security Definition. In conventional security definitions (for both second and first level ciphertexts) of a PRE scheme, it is common to let an adversary choose the challenge public key pk_{i^*} from a set $\mathcal{PK}$ of honest users' public keys, given to the adversary at the beginning of the security games. However, such a security definition for an encryption scheme is typically (polynomially) equivalent to one in which the challenge key is chosen by the challenger from the set $\mathcal{PK}$. Although this is not always true depending on the winning condition of the game (especially, in a security model where the winning condition of an adversary is affected by the queries made by the adversary), it is true for our second and first level security definitions. Moreover, note that in our security game for a first level ciphertext, there is no restriction on the re-encryption key generation query, and thus an adversary can freely request a re-encryption key from the challenge key pk_{i^*} to a corrupt key pk. It is actually possible to consider a "conventional" security definition for a first level ciphertext in which there is a set $\mathcal{PK}$ of honest users' public keys and an adversary can

freely choose the challenge key from $\mathcal{PK}$, and then to show that security in such a conventional model is (polynomially) equivalent to security in our definition.

The obvious advantage of these "fixed challenge key" style security definitions is that it makes the security analysis simpler, and therefore we adopt these definitions.

Second Level Decryption Queries. Previous works have concluded that since adversary A can easily convert a second level ciphertext into a first level ciphertext by using the re-encryption keys given to him, it is not necessary to consider second decryption queries. However, such an observation is incorrect. Concretely, we can construct a PRE scheme Π', which is secure if adversary A is not allowed to make second decryption queries, but which is insecure if A is allowed to make these, using a secure PRE scheme Π (in the sense of our definition) as a building block.

- The second level encryption algorithm for Π' first runs the second level encryption algorithm for Π, generating a second level ciphertext $\tilde{c}$, and outputs $c = (\tilde{c}||0)$ (i.e., 0 is attached).
- The second level decryption algorithm $\mathtt{Dec}_2$ for Π' ignores the last bit of the second level ciphertext $(\tilde{c}||0)$, and decrypts $\tilde{c}$ with the underlying second level decryption algorithm Π.
- The re-encryption algorithm rejects a ciphertext c if it is of the form $c = (\widetilde{c}||1)$, and otherwise ignores the last bit and re-encrypts $\widetilde{c}$ with the underlying re-encryption algorithm.
- The other algorithms for Π' are the same as those for Π.

With a scheme constructed as above, it is clear that adversary A can break second level CCA-SUPRE security by using the following second level decryption query. After A has received the second level challenge ciphertext $c_{i^*} = (\tilde{c}_{i^*}||0)$, A obtains $c'_{i^*} = (\tilde{c}_{i^*}||1)$ by inverting the least significant bit, and then makes a second level decryption query $(\mathsf{pk}_{i^*}, c'_{i^*})$ and receives m_b. On the other hand, if A is not allowed to make second level decryption queries, the security of the underlying scheme Π guarantees that Π' is secure. Therefore, in order to define CCA security as strongly as possible, we allow A to make both first and second level decryption queries in the second level security game.

Omitting Direct First Level Encryption Algorithm. The other difference between our security definition and [13] is that we do not include a first level encryption algorithm to generate a first level ciphertext in the syntax of a SUPRE scheme. This is simply because if we would like to have it in a SUPRE scheme, it can be implemented with an independent IND-CCA secure PKE scheme. This simplifies the syntax of the SUPRE scheme.

4 Resplittable Threshold Public Key Encryption

In this section, we introduce a new variation of a TPKE scheme, which we call *resplittable threshold public key encryption*, which is used as the main building block in our generic construction of SUPRE in the next section.

4.1 Resplittability in Threshold Public Key Encryption

TPKE [11] is an extension of PKE, where we consider a model with multiple-decryption servers, each of which holds one "secret key share." When a ciphertext c needs to be decrypted, it is sent to all the decryption servers. After receiving a partially decrypted ciphertext from each server, a combiner can then reconstruct the message in c if at least t shares are valid. Here t is called the threshold of the TPKE scheme. Our generic construction of a SUPRE scheme is based on a variant of TPKE, which we call *resplittable* TPKE.

Informally speaking, a resplittable TPKE scheme is a threshold encryption scheme with an additional randomized algorithm `TSplit`, which splits a secret key tsk into shares $tsk_1, tsk_2, \ldots$, in such a way that the TPKE scheme remains secure as long as the number of corrupted secret key shares *output by one execution of the* `TSplit` *algorithm* is less than the threshold t.

In fact, splittability of a secret key is an inherent functionality of an ordinary TPKE scheme, since TPKE usually requires a distributed key generation protocol, or a trusted server generates random secret key shares for each decryption server. The traditional TPKE scheme requires such splitting only once (in the key generation), whereas in order to use a TPKE scheme in our construction we require that a TPKE scheme be secure even after the polynomial number splitting. Although the requirement might look strong, as we shall see, we have a concrete efficient TPKE scheme that is resplittable.

Syntax. A resplittable TPKE scheme consists of the following six algorithms (`TKG`, `TEnc`, `TSplit`, `TShDec`, `TShVer`, `TCom`).

`TKG` takes as input a security parameter Λ, n and t, and outputs a secret key tsk and a public key tpk, denoted as $(tsk, tpk) \leftarrow \mathtt{TKG}(\Lambda, n, t)$.

`TEnc` takes as input a public key tpk and a message m, and outputs a ciphertext c, denoted as $c \leftarrow \mathtt{TEnc}(tpk, m)$.

`TSplit` takes as input a secret key tsk of the TPKE, outputs n shares of tsk and a verification key tvk, denoted as $(tsk_1, \ldots, tsk_n, tvk) \leftarrow \mathtt{TSplit}(tsk)$.

`TShDec` takes as input a public key tpk, a secret key share tsk_i $(1 \leq i \leq n)$ output by `TSplit` and a ciphertext c, and outputs a decryption share μ_i or a special symbol "$\perp$"(μ is an invalid share), denoted as $\mu_i \leftarrow \mathtt{TShDec}(tpk, tsk_i, c)$. Here, we assumed that `TShDec` is deterministic.

`TShVer` takes as input a public key tpk, a verification key tvk, a ciphertext c, an index i and a decryption share μ and outputs valid if μ is valid, or invalid if μ is invalid. When the output is valid, we say that μ is a valid decryption share of the ciphertext c.

`TCom` takes as input a public key tpk, a ciphertext c, and t decryption shares from different decryption servers and outputs a plaintext m (or a special symbol $\perp$ indicating "`invalid ciphertext`"), denoted as $m \leftarrow \mathtt{TCom}(tpk, tvk, c, \{\mu_1, \ldots, \mu_t\})$.

Correctness. For any $(tsk, tpk) \leftarrow \mathtt{TKG}(\Lambda, n, t)$ and any $(tsk_1, \ldots, tsk_n, tvk) \leftarrow \mathtt{TSplit}(tsk)$, we require the following two correctness properties: (1)For any

ciphertext c, if $\mu \leftarrow \mathtt{TShDec}(tpk, tsk_i, c)$, then $\mathtt{TShVer}(tpk, tvk, c, i, \mu)$ outputs valid. (2)If c is output from $\mathtt{TEnc}(tpk, m)$ and $S = \{\mu_{s_1}, \ldots, \mu_{s_t}\}$ is a set of decryption shares $\mu_{s_i} \leftarrow \mathtt{TShDec}(tpk, tsk_{s_i}, c)$ under t distinct secret key shares, then we require that $\mathtt{TCom}(tpk, tvk, c, S)$ output m.

Chosen Ciphertext Security. A (t, n)-re-splittable TPKE scheme is one in which the algorithm TSplit always outputs exactly n shares, and a ciphertext c together with any subset, smaller than t, of decryption shares of c do not leak essential information of the plaintext in c. We define chosen-ciphertext security of re-splittable TPKE based on the definition of CCA security for ordinary TPKE schemes [6,2]. Our definition is a natural extension of that in [6,2]. In particular, we only add "Split&Corruption query" (which is formally described below) in the attack model.

It is required that a polynomially-bounded adversary should not have any knowledge of the plaintext even given a decryption oracle, as long as the number of corrupt decryption servers is less than t under the same splitting of tsk.

Setup. First, the challenger runs $(tsk, tpk) \leftarrow \mathtt{TKG}(\Lambda, n, t)$ and gives tpk to A.

Split&Corruption Query. For the j-th split&corruption query $S = \{s_1, \ldots, s_{t-1}\}$, the challenger computes $(tsk_{j.1}, \ldots, tsk_{j.n}, tvk_j) \leftarrow \mathtt{TSplit}(tsk)$ and returns $(tsk_{j.s_1}, \ldots, tsk_{j.s_{t-1}}, tvk_j)$ to A. The challenger stores $\{tsk_{j.i}\}_{i \in \{1,\ldots,n\}}$ and tvk_j for later share decryption queries from A.

Share Decryption Query. For a share decryption query (tvk_j, i, c), where tvk is required to be one of the answers to previously asked split&corruption queries, i is an index of a decryption server, and $c \neq c^*$ is a ciphertext, the challenger finds $tsk_{j,i}$ that is previously generated, and returns a partial decryption result $\mu_i \leftarrow \mathtt{TShDec}(tpk, tsk_{j.i}, c)$ to A.

Challenge. This query is asked only once. For a challenge query (m_0, m_1), the challenger picks the challenge bit $b \in \{0, 1\}$ and returns $c^* \leftarrow \mathtt{TEnc}(tpk, m_b)$ to A.

Finally, A outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$. We define the advantage of A by $\mathrm{Adv}^{\mathrm{CCA-TPKE}}_{(\mathsf{A},n,t)}(\Lambda) = |\Pr[b = b'] - \frac{1}{2}|$.

Decryption Consistency. We define the decryption consistency of resplittable TPKE based on [6]. Consistency of decryption is defined using the following game: The game starts with the **Setup** and several queries as in the game above, except the challenge query. The adversary then outputs a ciphertext c, a verification key tvk, and two sets of decryption shares $S = \{\mu_1, \ldots, \mu_t\}$ and $S' = \{\mu'_1, \ldots, \mu'_t\}$ such that $|S| = |S'| = t$.

The adversary wins if

(a) tvk is one of verification keys returned as a response to the adversary A's split&corruption query.
(b) S and S' are valid decryption shares for a ciphertext c under tvk;
(c) S and S' contain decryption shares from t distinct servers; and
(d) $\mathtt{TCom}(tpk, tvk, c, S) \neq \mathtt{TCom}(tpk, tvk, c, S')$.

We let $\mathrm{Adv}^{\mathrm{DC}}_{(\mathsf{A},n,t)}(\Lambda)$ denote the adversary's advantage in winning this game.

Definition 4. *We say that a TPKE scheme is secure if for any polynomial n and t where $0 < t \leq n$, and any probabilistic polynomial adversary A, the functions $\mathrm{Adv}^{\mathrm{CCA-TPKE}}_{(\mathsf{A},n,t)}(\Lambda)$ and $\mathrm{Adv}^{\mathrm{DC}}_{(\mathsf{A},n,t)}(\Lambda)$ are negligible.*

$\mathtt{TKG}(\Lambda)$: $x, y, z \leftarrow \mathbb{Z}_p$ $g_1 = g^x, g_2 = g^y, h_1 = g^z$ Let $tsk = x$ and $tpk = (p, \mathbb{G}, \mathbb{G}_1, g, g_1, g_2, h_1)$ Return tsk, tpk	$\mathtt{TEnc}(tpk, m)$ $(sk, vk) \leftarrow \mathtt{SKG}(\Lambda)$ $r \leftarrow \mathbb{Z}_p$ $c' = (C, D, E)$ $= (g^r, (g_1^{vk} h_1)^r, m \cdot e(g_1, g_2)^r)$ $\sigma \leftarrow \mathtt{Sign}(sk, c')$ Return $c = (c', vk, \sigma)$
$\mathtt{TSplit}(tsk)$ $f \leftarrow \mathbb{Z}_p[X]$ such that $deg(f) = t - 1$ and $f(0) = x$ Define $tsk_i = f(i)$ and $tvk = (g^{f(1)}, \ldots, g^{f(n)})$ Return $tsk_1, \ldots, tsk_n, tvk$	$\mathtt{TShDec}(tpk, tsk_i, c)$ If invalid $\leftarrow \mathtt{SVer}(vk, c', \sigma)$, return $\perp$ If $e(C, g_1^t h_1) \neq e(D, g)$, return $\perp$. Otherwise, return $\mu_i = C_i = C^{f(i)}$.
$\mathtt{TShVer}(tpk, tvk, c, i, \mu)$ If invalid $\leftarrow \mathtt{SVer}(vk, c', \sigma)$, return invalid. If $e(C_i, g) \neq e(C, g^{f(i)})$, return invalid Otherwise, return valid.	$\mathtt{TCom}(tpk, tvk, c, S = \{\mu_{s_1}, \ldots, \mu_{s_t}\})$ If invalid $\leftarrow \mathtt{TShVer}(tpk, tvk, c, i, \mu_{s_i})$, return $\perp$. Else, compute $m = E/e(\prod_{i=1}^{t} C_i^{\lambda_i}, g_2)$ using Lagrange coefficients λ_i satisfying $f(0) = \Sigma_{i=1}^{t} \lambda_i f(i)$ Return m.

Fig. 1. Concrete Resplittable TPKE Scheme Based on [2]

4.2 Concrete Resplittable Threshold Public Key Encryption

In this paper, we give Arita and Tsurudome's scheme [2], which is based on Boyen et al.'s PKE scheme [7], as an example of a resplittable TPKE scheme. Arita and Tsurudome proposed a conversion from any stag-CCA-secure threshold tag-based encryption scheme to a CCA-secure TPKE scheme. Moreover, they proposed two concrete stag-CCA-secure threshold tag-based encryption schemes, one of which is based on the decisional bilinear Diffie-Hellman (DBDH) assumption and the other one is based on the decisional linear assumption. In this subsection, we show that Arita and Tsurudome's TPKE scheme under the DBDH assumption supports key-resplittability.

Here, let $\mathbb{G}$ be a group of prime order p with generator g, $\mathbb{G}_1$ be a group of prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map. If a bilinear Diffie-Hellman tuple $(g, g^a, g^b, g^c, e(g,g)^{abc})$ is indistinguishable from $(g, g^a, g^b, g^c, e(g,g)^d)$ where d is chosen from $\mathbb{Z}_p$ randomly, we say that the DBDH assumption holds. We define $\mathrm{Adv}^{\mathrm{dbdh}}_{\mathsf{A}}(\Lambda) = |\Pr[1 \leftarrow \mathsf{A}(g, g^a, g^b, g^c, e(g,g)^{abc})] - \Pr[1 \leftarrow \mathsf{A}(g, g^a, g^b, g^c, e(g,g)^d)]|$ where $a, b, c, d \in \mathbb{Z}_p$ are chosen randomly.

Let (SKG, Sign, SVer) be a strong one-time signature scheme. Then, we construct a concrete resplittable PKE scheme as in Fig. 1.

Theorem 1. *If the DBDH assumption holds and the one-time signature is strongly unforgeable, the TPKE scheme in Fig. 1 is a secure resplittable TPKE scheme.*

The proof is omitted due to lack of space, however it is easily inferred from the proof given in [2]. We describe the full proof in the full version of this paper.

5 Generic Construction of SUPRE Based on TPKE

In this section, we propose a generic construction for a SUPRE scheme, and prove its CCA security. Our construction of SUPRE is based on a $(2, 2)$-re-splittable TPKE scheme (TKG, TEnc, TSplit, TShDec, TShVer, TCom), a PKE scheme (PKG, PEnc, PDec), and a signature scheme (SKG, Sign, SVer). Using these building blocks, we construct a SUPRE scheme as in Fig. 2.

Construction Ideas. The re-splittability of the building block (2,2)-re-splittable TPKE scheme plays a central role in the main functionality of a SUPRE scheme, that is, re-encryption key generation and re-encryption. The main components of a re-encryption key $rk_{i\to j}$ from user i with key-pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ to user j with key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ are $tsk_{i.2}$ and ψ, where ψ is an encryption of $tsk_{i.1}$ under the PKE public key pk_j contained in pk_j, and $tsk_{i.1}$ and $tsk_{i.2}$ are secret key shares that are (re-)split using the TPKE secret key tsk_i contained in sk_i. When a proxy with $rk_{i\to j}$ re-encrypts a second level (TPKE) ciphertext c_i, it first calculates a share-decryption μ_2 of c using $tsk_{i.2}$, which is directly contained in $rk_{i\to j}$, and rejects the ciphertext c_i if $\mu_2 = \perp$. (Note that since we are using a (2,2)-TPKE scheme, if one of the decryption shares of ciphertext c is $\perp$, it must mean that c is an invalid ciphertext.) Otherwise, the proxy "wraps" (c_i, μ_2, ψ) using the building PKE scheme. That is, the proxy generates a re-encrypted (first level) ciphertext $\widehat{c}_j$ by encrypting (c_i, μ_2, ψ) under the PKE public key $\widehat{pk}_j$ contained in pk_j. Then, user j who owns the PKE decryption keys $\widehat{dk}_j$ and dk_j corresponding to $\widehat{pk}_j$ and pk_j, respectively, can decrypt $\widehat{c}_j$ and ψ. Since $\widehat{c}_j$ contains (c_i, μ_2, ψ), and ψ contains $tsk_{i.1}$, after decrypting $\widehat{c}_j$ using $\widehat{dk}_j$, user j can share-decrypt c using $tsk_{i.1}$ to obtain μ_1, and then recover the plaintext m from the shares (μ_1, μ_2) by using the combination algorithm of the TPKE scheme.

Intuitively, CCA security of the building block PKE scheme prevents a malicious proxy with $rk_{i\to j}$ from learning $tsk_{i.1}$ from ψ or modifying ψ in any meaningful way. Moreover, security of the (2,2)-re-splittable TPKE scheme guarantees that despite having one decryption share $tsk_{i.2}$, it can essentially learn nothing about the contents of a second level ciphertext. Furthermore, owing to the "wrapping" in the re-encryption process, CCA security of the building block PKE scheme also guarantees that a correctly re-encrypted ciphertext $\widehat{c}$ leaks no

information and has non-malleability, leading directly to CCA security for first level ciphertexts.

However, there are several other subtle technical points that need to be taken into account to make our scheme CCA secure (especially, second level CCA-SUPRE secure).

- We include the public keys $(\mathsf{pk}_i, \mathsf{pk}_j)$ of the "source" user i and the "destination" user j in $rk_{i\to j}$. These keys are then included in a first level ciphertext $\widehat{c}$ re-encrypted using $rk_{i\to j}$, and the equality of the keys is checked in the first level decryption algorithm. This ensures that $rk_{i\to j}$ and $\widehat{c}$ are only valid under a particular pair of source user i and destination user j, and prevents the proposed scheme from being vulnerable to "key substitution"-like attacks.
- We also include the verification key tvk_i (of the TPKE scheme) corresponding to the secret key shares $(tsk_{i.1}, tsk_{i.2})$ in $rk_{i\to j}$, and it is also included in a first level ciphertext re-encrypted using $rk_{i\to j}$. This allows the destination user j to be convinced (in the first level decryption algorithm) that the proxy has actually made a valid decryption share μ_2. Decryption consistency, together with correctness of the re-splittable TPKE scheme, guarantees that the plaintext m recovered in the first level decryption algorithm is actually the same as that of the original second level ciphertext.
- We "wrap" every component other than $tsk_{i.2}$ in a re-encryption key $rk_{i\to j}$ using a signature with signing key sk_i, which is contained in user i's secret key sk_i. ($tsk_{i.2}$ is not signed since it does not appear in the plaintext of re-encrypted ciphertext $\widehat{c}$.) This signature ensures that the re-encryption key $rk_{i\to j}$ is "non-malleable", so that the destination user j can be convinced that the first level ciphertext $\widehat{c}$ received has actually been calculated using a correctly generated re-encryption key $rk_{i\to j}$.

These ideas make it possible to prove that our scheme is CCA secure.

Security. The security of our generic construction of SUPRE is guaranteed by the following theorems.

Theorem 2. *If the PKE scheme is IND-CCA secure in the multi-user setting, the signature scheme is strongly unforgeable, and the resplittable TPKE scheme is secure, then our proposed PRE scheme is second level CCA-SUPRE secure.*

We will describe the full proof of the above theorem in the full version of this paper.

Theorem 3. *If the PKE scheme is IND-CCA secure, our generic construction is first level CCA-SUPRE secure.*

Proof. Assume towards a contradiction that there exists a first level SUPRE-CCA adversary A such that $Adv_{\mathsf{A}}^{first}(\Lambda)$ is not negligible. Then we show that we can use A to construct another adversary B that has non-negligible IND-CCA advantage regarding the building block PKE scheme. The construction of B is as follows. First, B is given $\widehat{pk}_{i^*}$. B generates the challenge public/secret

$\mathtt{KG}(\Lambda)$:
$(tsk, tpk) \leftarrow \mathtt{TKG}(\Lambda, 2, 2)$
$(\widehat{dk}, \widehat{pk}) \leftarrow \mathtt{PKG}(\Lambda)$, $(dk, pk) \leftarrow \mathtt{PKG}(\Lambda)$, $(sk, vk) \leftarrow \mathtt{SKG}(\Lambda)$
Let $\mathsf{sk} = (tsk, \widehat{dk}, dk, sk)$,$\mathsf{pk} = (tpk, \widehat{pk}, pk, vk)$
Return sk, pk

$\mathtt{RKG}(\mathsf{sk}_i, \mathsf{pk}_j)$
Parse $\mathsf{sk}_i = (tsk_i, \widehat{dk}_i, dk_i, sk_i)$. Parse $\mathsf{pk}_j = (tpk_j, \widehat{pk}_j, pk_j, vk_i)$
$(tsk_{i.1}, tsk_{i.2}, tvk_i) \leftarrow \mathtt{TSplit}(tsk_i)$
$\psi \leftarrow \mathtt{PEnc}(pk_j, tsk_{i.1})$
$\sigma \leftarrow \mathtt{Sign}(sk_i, \langle \psi || tvk_i || \mathsf{pk}_i || \mathsf{pk}_j \rangle)$
Return $rk_{i \to j} = (\mathsf{pk}_i, \mathsf{pk}_j, tsk_{i.2}, \psi, tvk_i, \sigma)$

$\mathtt{Enc}(\mathsf{pk}_i, m)$
Parse $\mathsf{pk}_i = (tpk_i, \widehat{pk}_i, pk_i, vk_i)$
Return $c \leftarrow \mathtt{TEnc}(tpk_i, m)$

$\mathtt{REnc}(rk_{i \to j}, c_i)$
Parse $rk_{i \to j} = (\mathsf{pk}_i, \mathsf{pk}_j, tsk_{i.2}, \psi, tvk_i, \sigma)$
If $\mathtt{SVer}(vk_i, \langle \psi || tvk_i || \mathsf{pk}_i || \mathsf{pk}_j \rangle, \sigma) = \mathsf{invalid}$, return $\perp$.
$\mu_2 \leftarrow \mathtt{TShDec}(tpk_i, tsk_{i.2}, c_i)$
If $\mu_2 = \perp$, return $\perp$.
Return $\widehat{c}_j \leftarrow \mathtt{PEnc}(\widehat{pk}_j, \langle \mathsf{pk}_i || \mathsf{pk}_j || c_i || \mu_2 || \psi || tvk_i || \sigma \rangle)$

$\mathtt{Dec}_1(\mathsf{sk}_j, \widehat{c}_j)$
Parse $\mathsf{pk}_j = (tpk_j, \widehat{pk}_j, pk_j, vk_i)$. Parse $\mathsf{sk}_j = (tsk_j, \widehat{dk}_j, dk_j, sk_i)$
$\langle \mathsf{pk}'_i || \mathsf{pk}'_j || c_i || \mu_2 || \psi || tvk_i || \sigma \rangle \leftarrow \mathtt{PDec}(\widehat{dk}_j, \widehat{c}_j)$
If the result is $\perp$ or $\mathsf{pk}'_j \neq \mathsf{pk}_j$, return $\perp$.
If $\mathtt{SVer}(vk_i, \langle \psi || tvk_i || \mathsf{pk}'_i || \mathsf{pk}'_j \rangle, \sigma) = \mathsf{invalid}$, return $\perp$.
$tsk_{i.1} \leftarrow \mathtt{PDec}(dk_j, \psi)$
If $tsk_{i.1} = \perp$, return $\perp$.
$\mu_1 \leftarrow \mathtt{TShDec}(tpk_i, tsk_{i.1}, c_i)$
If $\mu_1 = \perp$, return $\perp$.
If $\mathtt{TShVer}(tpk_i, tvk_i, c_i, 2, \mu_2) = \mathsf{invalid}$, return $\perp$.
Return $m \leftarrow \mathtt{TCom}(tpk_i, tvk_i, c_i, \{\mu_1, \mu_2\})$

$\mathtt{Dec}_2(\mathsf{sk}_i, c)$
Parse $\mathsf{pk}_i = (tpk_i, \widehat{pk}_i, pk_i, vk_i)$. Parse $\mathsf{sk}_i = (tsk_i, \widehat{dk}_i, dk_i, sk_i)$
$(tsk_{i.1}, tsk_{i.2}, tvk_i) \leftarrow \mathtt{TSplit}(tsk_i)$
$\mu_1 \leftarrow \mathtt{TShDec}(tpk_i, tsk_{i.1}, c)$
If $\mu_1 = \perp$, return $\perp$.
$\mu_2 \leftarrow \mathtt{TShDec}(tpk_i, tsk_{i.2}, c)$
If $\mu_2 = \perp$, return $\perp$.
Return $m \leftarrow \mathtt{TCom}(tpk_i, tvk_i, c, \{\mu_1, \mu_2\})$

Fig. 2. Our Generic Construction of SUPRE

key except $\widehat{dk}_{i^*}$, and then gives the challenge public key pk_{i^*} to A. Now, since B knows tsk_{i^*}, dk_{i^*} and sk_{i^*}, B can simulate re-encryption key generation, re-encryption, and second level decryption queries perfectly. Also, B can simulate first decryption queries using decryption queries on $\widehat{pk}_{i^*}$ and $(tsk_{i^*}, dk_{i^*}, sk_{i^*})$ that B knows. When A submits two plaintexts (m_0, m_1) of equal length and a key

pair $(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$, B proceeds as follows: (1) Parse sk_A as $(tsk_\mathsf{A}, \widehat{dk}_\mathsf{A}, dk_\mathsf{A}, sk_\mathsf{A})$ and pk_A as $(tpk_\mathsf{A}, \widehat{pk}_\mathsf{A}, pk_\mathsf{A}, vk_\mathsf{A})$. (2) Execute $c_\gamma \leftarrow \mathtt{TEnc}(tpk_\mathsf{A}, m_\gamma)$ for $\gamma \in \{0,1\}$. (3) Execute $(tsk_{\mathsf{A}.1}, tsk_{\mathsf{A}.2}, tvk_\mathsf{A}) \leftarrow \mathtt{TSplit}(tsk_\mathsf{A})$, $\psi \leftarrow \mathtt{PEnc}(pk_{i^*}, tsk_{\mathsf{A}.1})$, and $\sigma \leftarrow \mathtt{Sign}(sk_{i^*}, \langle \psi || tvk_\mathsf{A} | \mathsf{pk}_\mathsf{A} || \mathsf{pk}_{i^*} \rangle)$. (4) Execute $\mu_{2.\gamma} \leftarrow \mathtt{TShDec}(tpk_\mathsf{A}, tsk_{\mathsf{A}.2}, c_0)$ for $\gamma \in \{0,1\}$. (5) Set $M_\gamma = \langle \mathsf{pk}_\mathsf{A} || \mathsf{pk}_{i^*} || c_\gamma || \mu_{2,\gamma} || \psi || tvk_\mathsf{A} || \sigma \rangle$ for $\gamma \in \{0,1\}$, submit (M_0, M_1) to the challenger, and receive (B's) challenge ciphertext $\widehat{c}_{i^*}$. (6) Return $\widehat{c}_{i^*}$ to A as A's challenge ciphertext. Finally, when A terminates with a guess bit b', B outputs this b' and terminates. Since B perfectly simulates the first level CCA-SUPRE game for A, B's advantage is exactly $\mathrm{Adv}_\mathsf{A}^{first}(\Lambda)$, which is not negligible. This contradicts our assumption that the PKE scheme is IND-CCA secure, and thus the theorem follows. □

Acknowledgements. We would like to thank the anonymous CT-RSA reviewers for their helpful comments. Yutaka Kawai and Takahiro Matsuda are supported by Research Fellowships of the Japan Society for the Promotion of Science for Young Scientists. Jian Weng was supported by the National Science Foundation of China under Grant Nos. 60903178 and 61133014, the Fundamental Research Funds for the Central Universities under Grant No. 21610204, and the Guangdong Provincial Science and Technology Project under Grand No. 2010A032000002. Yunlei Zhao is partly supported by a grant from the Major State Basic Research Development (973) Program of China (No. 2007CB807901) and a grant from the National Natural Science Foundation of China NSFC (No. 61070248).

References

1. An, J.H., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
2. Arita, S., Tsurudome, K.: Construction of Threshold Public-Key Encryptions through Tag-Based Encryptions. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 186–200. Springer, Heidelberg (2009)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. ACM Trans. Inf. Syst. Secur. 9(1), 1–30 (2006)
4. 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)
5. 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)
6. Boneh, D., Boyen, X., Halevi, S.: Chosen Ciphertext Secure Public Key Threshold Encryption Without Random Oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)
7. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security From Identity-Based Techniques. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 320–329 (2005)

8. Canetti, R., Goldwasser, S.: An Efficient {\it Threshold} Public Key Cryptosystem Secure against Adaptive Chosen Ciphertext Attack. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90–106. Springer, Heidelberg (1999)
9. Canetti, R., Hohenberger, S.: Chosen-Ciphertext Secure Proxy Re-encryption. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 185–194 (2007)
10. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient Unidirectional Proxy Re-Encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (2010)
11. Desmedt, Y., Frankel, Y.: Threshold Cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
12. Ivan, A.-A., Dodis, Y.: Proxy Cryptography Revisited. In: NDSS (2003)
13. Libert, B., Vergnaud, D.: Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
14. Mambo, M., Okamoto, E.: Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences E80-A(1), 54–63 (1997)
15. Matsuda, T., Nishimaki, R., Tanaka, K.: CCA Proxy Re-Encryption without Bilinear Maps in the Standard Model. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 261–278. Springer, Heidelberg (2010)
16. Rackoff, C., Simon, D.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
17. Shao, J., Cao, Z.: CCA-Secure Proxy Re-encryption without Pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)
18. Shao, J., Cao, Z., Liu, P.: CCA-Secure PRE Scheme without Random Oracles. Cryptology ePrint Archive, Report 2010/112 (2010), http://eprint.iacr.org/
19. Shao, J., Liu, P.: CCA-Secure PRE Scheme without Public Verifiability. Cryptology ePrint Archive, Report 2010/357 (2010), http://eprint.iacr.org/
20. Shoup, V., Gennaro, R.: Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)
21. Weng, J., Zhao, Y., Hanaoka, G.: On the Security of a Bidirectional Proxy Re-encryption Scheme from PKC 2010. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 284–295. Springer, Heidelberg (2011)
22. Zhang, X., Chen, M., Li, X.: Comments on Shao-Cao's Unidirectional Proxy Re-Encryption Scheme from PKC 2009. Cryptology ePrint Archive, Report 2009/344 (2009)

A New Difference Method for Side-Channel Analysis with High-Dimensional Leakage Models

Annelie Heuser[1,4], Michael Kasper[2,4],
Werner Schindler[3,4], and Marc Stöttinger[1,4]

[1] Darmstadt University of Technology, Germany
{Heuser,Stoettinger}@iss.tu-darmstadt.de
[2] Fraunhofer Institute for Secure Information Technology (SIT), Germany
Michael.Kasper@sit.fraunhofer.de
[3] Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany
Werner.Schindler@bsi.bund.de
[4] Center for Advanced Security Research Darmstadt (CASED), Germany

Abstract. The goal of the DPA contest v2 (2009 – 2010) was to find the most efficient side-channel attack against a particular unprotected AES-128 hardware implementation. In this paper we discuss two problems of general importance that affect the success rate of profiling based attacks, and we provide effective solutions. First, we consider the impact of temperature variations on the power consumption, which causes a so-called *drifting offset*. To cope with this problem we introduce a new method called *Offset Tolerant Method (OTM)* and adjust OTM to the stochastic approach (SA-OTM). The second important issue of this paper concerns the choice of an appropriate leakage model as this determines the success rate of SA and SA-OTM. Experiments with *high-dimensional* leakage models show that the overall leakage is not only caused by independent transitions of bit lines. Compared to the formely best submitted attack of the DPA contest v2 the combination of SA-OTM with high-dimensional leakage models reduces the required number of power traces to 50%.

Keywords: Side-Channel Analysis, Stochastic Approach, Environmental Influences, Drifting Offset, High-dimensional Leakage Models.

1 Introduction

For more than a decade side-channel analysis has been an important field of research in both academia and industry. Usually these attacks apply mathematical techniques, e.g., statistical methods, to exploit compromising side-channel leakage (e.g., runtime behavior, power consumption or electromagnetic emanation), which is emitted during the regular execution of a cryptographic algorithm. Power attacks can be divided into non-profiled and profiled methods. Prominent representatives of non-profiled side-channel attacks are Differential Power Analysis [11], Correlation Power Analysis [2], and Mutual Information Analysis [6]. These attacks try to recover the secret information without a preceding profiling

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 365–382, 2012.

phase. Profiled side-channel attacks, such as template attacks [3] or the stochastic approach [17], have the potential to be much more powerful and efficient. In a profiling based attack an adversary (attacker, designer, evaluator) uses a training device to characterize the leakage of a cryptographic implementation by creating templates or by developing a well-fitted leakage model. Then he tries to recover the key from the target device, using the knowledge from the profiling phase.

Generally speaking, measurements performed by different laboratories are often difficult to compare due to different acquisition platform sensitivities, different implementations of cryptographic algorithms, noise and other environmental influences. The organizers of the DPA contest v2 [4] provided measurement traces that allow a fair comparison of several side-channel attacks. We decided to apply the stochastic approach.

In this contribution we deal with the two important problems that may affect the success rate of profiling based attacks. First, we highlight difficulties that arise from environmental influences during the acquisition phase. Motivated by the DPA contest v2 measurements we investigate the impact of temperature variations. In fact, variations of the environmental temperature as well as temperature variations inside the device may change the (average) level of power consumption and thus the level of electrical current and voltage consumption. We denote this unexpected phenomenon as *drifting offset*. The origin of temperature variations, their impact on the power and current consumption, and possible preventive measures are discussed in Sect. 2. In Sect. 4 we introduce a new algorithmic method, which we denote as *Offset Tolerant Method* (OTM) and integrate it into the stochastic approach, abbreviated by SA-OTM.

Second, we consider the precise representation of the compromising side-channel leakage by suitable leakage models. Profiling-based attacks are very powerful and effective tools but their efficiency strongly depends on the suitability of the applied leakage model. As stated in [13] several formal works assume that independent transitions of bit lines imply independent contributions of side-channel leakage. If this assumption is valid a leakage model that only considers the input/output bits of the SBox separately will be sufficient. However, Renauld et al. [13] uses Mutual Information Analysis as an information theoretic metric [18] to show that this assumption may not always be valid in practice. With regard to this observation we apply different *high-dimensional* leakage models, which represent the individual leakage of each bit line as well as the leakage caused by the combination of several bit lines. Referred to the DPA contest v2 the combination of SA-OTM with high-dimensional leakage models results in the best success rates.

2 Extrinsic and Environmental Influences on Side-Channel Evaluation Process

It is well-known that extrinsic and environmental influences as temperature, cosmic radiation and terrestrial radiation have an impact on the design in terms of

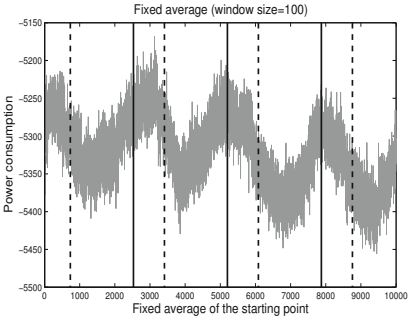


Fig. 1. Fixed average of the starting point; points in time: solid line 04:05 pm (starting time + 24h), dotted line 0:00 (midnight + 24h)

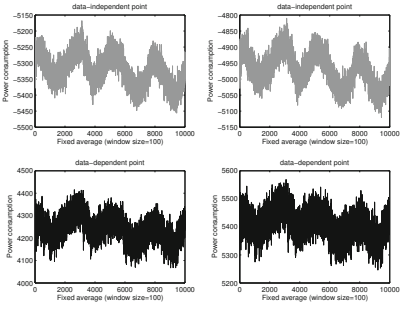


Fig. 2. Drifting offset at data-dependent and data-independent time points

the reliability and dependability of the integrated circuits functionality. However, the relevance of these phenomena in security analysis and, in particular in the side-channel analysis, have not been subject of public discussions yet. Usually, it is tacitly assumed that the power traces are recorded under constant environmental conditions. Neither temperature changes nor explicit influences caused by variations in the temperature of a system state are considered. We found out that the power curves of the DPA contest v2 [4] show a *drifting offset*, which might result from temperature variations. The template base of the DPA contest v2 consists of 1.000.000 traces, which were recorded during approximately 3 days and 19 hours. To give evidence for the drifting offset we selected particular time instants for all power traces and calculated the mean value over non-overlapping sets of 100 subsequent traces, which gave 10.000 mean values. Figure. 1 shows the mean power values at the starting point of the power traces. The dotted line corresponds to the beginning of a new day, and the solid line marks a 24 hour cycle. Figure. 1 illustrates the correspondence of the mean power consumption to the diurnal rhythm. Figure 2 shows the drifting offset for data-independent time points where no encryption is performed (gray) and for data-dependent points (black). Obviously, the drifting offset is larger than (average) effects that stem from data-dependent computations.

In order to confirm that the environmental temperature is the true reason for the existence of drifting offsets we performed own measurements on the SASEBO-GII platform. We simulated environmental temperature variations by mounting a peltier element and a cooling / heating system on the surface of the target FPGA. Figure 3 shows the voltage drop over a measurement shunt V_M for a single time instant. This reveals the direct relation between the environmental temperature and the power consumption of a device, which is proportionally bounded to the measured voltage drop of the shunt in the ground line. Note that the voltage drop at trace $\approx$ 7500 results from the activation of the peltier

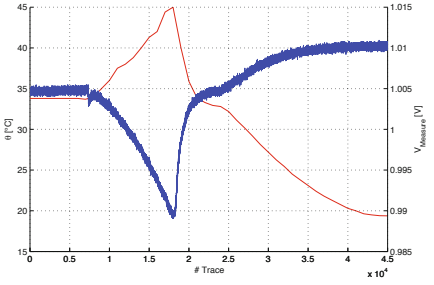


Fig. 3. Dependency between environmental temperature and power consumption. The thin red line represents the temperature while the thick blue line stands for the measured voltage drop.

element. Physical coherencies and possible preventive measures are discussed in the following.

3 Impact of Environmental Conditions

In the present section we analyze the impact of environmental conditions on the power consumption. We focus on the material specific temperature coefficient, denoted by α_{θ_0}, and on the impact to the characteristic ohmic resistance $\vartheta(\theta_0)$ of a target circuit. Eq. (1) provides a (linearized) formula that expresses the impact of the difference between the actual temperature θ and a reference temperature θ_0 on an ohmic resistor

$$R(\theta) = \vartheta(\theta_0) \cdot (1 + \alpha_{\theta_0} \cdot (\theta - \theta_0)) \,. \tag{1}$$

A measurement circuit usually consists of a target circuit (e.g., an FPGA configured with the cryptographic 'target' implementation) and a set of further electronic board components. The measurement circuit is usually realized by an ohmic shunt, which is chained between the target and a stable power supply. The voltage drop over this shunt V_M is used to calculate the power consumption of the target. The voltage divider (2) provides a simplified model for the relation between the supply voltage V_{cc} of the target device and V_M,

$$V_M = \frac{R_{board}}{R_{board} + R_{target}} \cdot V_{cc} \,, \tag{2}$$

where R_{target} denotes the ohmic resistance of target under attack and R_{board} denotes the overall resistance of all ohmic components of the above mentioned measurement circuit. Substituting Eq. (1) into the voltage divider for V_M gives Eq. (3)

$$V_M = \frac{V_{cc}}{1 + \left(\frac{\vartheta_{target}(\theta_{0,target})}{\vartheta_{board}(\theta_{0,board})}\right) \cdot \left(\frac{1+\alpha_{\theta_{0,target}} \cdot (\theta_{target} - \theta_{0,target})}{1+\alpha_{\theta_{0,board}} \cdot (\theta_{board} - \theta_{0,board})}\right)} \,. \tag{3}$$

Further experiments with the SASEBO-G II FPGA evaluation board verified that the impact of $\alpha_{\theta_{0,board}}$ is much smaller than the impact of $\alpha_{\theta_{0,target}}$. Consequently, the power consumption is more affected by temperature variations on the FPGA than by the temperature variations of the shunt and other board components. Hence Eq. (3) may be simplified to

$$V_M = \frac{V_{cc}}{1 + \left(\frac{\vartheta_{target}(\theta_{0,target})}{\vartheta_{board}(\theta_{0,S})}\right) \cdot \left(1 + \alpha_{\theta_{0,target}} \cdot \underbrace{(\theta_{target} - \theta_{0,target})}_{\substack{heating>0,\\ cooling<0}}\right)}. \tag{4}$$

For the SASEBO-G II FPGA evaluation board (as for similar evaluation boards) V_M increases significantly if the target FPGA is cooled down and decreases significantly if the FPGA is heated, which coincides with the expositions in Sect. 2.

Preventing drifting offsets by providing constant environmental conditions. An intuitive and natural method to prevent drifting offsets is to keep the temperature of both the device and the environment constant. The environmental temperature can be controlled when using a heating cabinet or a climatic chamber. The device may be preheated or cooled during the measurements in order to stabilize the temperature of the device.

However, these measures reduce the thermal effects only to a certain level. Since the thermal processes are very slow and the response time is very long it is yet difficult to control them precisely. Thus, the temperature gradient between the device and the environment should to be stable for a certain time interval. This time interval is certainly shorter than the full profiling measurement period. Moreover, the adversary may not have unlimited access to the target device so that these measures may not always be possible. This raises several questions for further research: If an attacker is able to learn on a training device of the same type under stable environmental conditions, can he also ensure constant conditions during the attack? Is it possible to enforce identical conditions in different situations? If not: how can unstable environmental conditions be handled efficiently?

4 A Novel Method for Effective Offset Elimination

Like for other attacks that (maybe implicitly) consider the average power consumption, e.g., template attacks, the efficiency of the stochastic approach may decrease significantly in presence of drifting offsets (Figs. 1 and 2). In the light of Sects. 2 and 3 we may assume that the offset drifts slowly. An intuitive approach to eliminate drifting offsets is to consider differences of consecutive power traces in place of the power traces themselves. This is an *Offset Tolerant Method* (OTM), and we adjust this method to the stochastic approach (SA), abbreviated by SA-OTM. This may sound simple, however, it will turn out later that several mathematical difficulties have to be overcome.

4.1 The 'Normal' Stochastic Approach: A Brief Summary

The 'normal' stochastic approach is an established, effective method in profiled power analysis, which combines engineer's knowledge and expertise with advanced stochastic methods [17,7,16,10,8]. In this subsection we summarize its central steps. In Subsection 4.2 we will refer to this description, and we work out the differences to SA-OTM. Principal component analysis (PCA) is well-known in the context of template attacks [1]. Below we adjust PCA to the stochastic approach. We begin with some notations.

Notation 1. We denote subkeys by $k \in \{0,1\}^s$ while $x \in \{0,1\}^p$ stands for (the relevant part of) the plaintext or ciphertext, respectively (typically, 8 or 16 bits). Random variables are denoted by capital letters, realizations thereof, i.e. values taken on by these random variables, by the corresponding small letters. Vectors are written in bold, e.g., $\mathbf{t}$ stands for $(t_1, \ldots, t_m)$, and $\mathbf{R_t}$ denotes the random vector $(R_{t_1}, \ldots, R_{t_m})$. Accordingly, $\mathbf{I_t}(x,k)$, $\mathbf{i_t}(x,k)$, $\mathbf{h}^*_{\mathbf{t};k}(x,k)$ etc. while $\sim$ indicates estimates. We write $\mathrm{diag_n}(\mathrm{b}_1, \ldots, \mathrm{b_n})$ for a diagonal $n \times n$ square matrix with diagonal elements $b_1, \ldots, b_n$, and $\mathcal{N}_n(\mu, F)$ denotes an n-dimensional normal distribution with mean vector μ and covariance matrix F. Finally, $f_F(\cdot)$ denotes the density of $N(0,F)$.

The stochastic approach refers to the mathematical model

$$I_t(x,k) = h_t(x,k) + R_t \tag{5}$$

where t denotes a time instant. The power consumption $i_t(x,k)$ is interpreted as a realization of a random variable $I_t(x,k)$ whose (unknown) distribution depends on the pair (x,k). The leakage function $h_{t;k}(x,k)$ quantifies its deterministic part, which depends on x and k, while R_t denotes the noise. W.l.o.g. we may assume $E(R_t) = 0$. Note that both the leakage function $h_t(\cdot,\cdot)$ and the distribution of the noise are unknown and thus have to be estimated.

Profiling Phase. Let $t \in \{t_1, \ldots, t_m\}$ and $k \in \{0,1\}^s$ be fixed for the moment. We view the restricted function $h_{t;k}\colon \{0,1\}^p \times \{k\} \to \mathbb{R}$, $h_{t;k}(x,k) := h_t(x,k)$ as an element of the 2^p-dimensional real vector space $\mathcal{F}_k := \{h'\colon \{0,1\}^p \to \mathbb{R}\}$. Basis functions $g_{0,j;k}(\cdot,k) = 1$ (constant function), $\ldots, g_{u-1,t;k}(\cdot,k)$ shall be selected under consideration of the concrete implementation, since they shall capture the relevant source of side-channel leakage (cf. e.g., [8] and Sect. 5). The SA does not aim at the exact function $h_{t;k}(\cdot,k)$ itself but at its best approximator $h^*_{t;k}(\cdot,k)$ in $\mathcal{F}_{u,t;k}$, the subspace which is spanned by $g_{0,j;k}(\cdot,k), \ldots, g_{u-1,t;k}(\cdot,k)$. Using the power measurements $i_t(x_1,k), \ldots, i_t(x_{N_1},k) \in \mathbb{R}$ the least square estimate $\widetilde{h}^*_{t;k}(\cdot,k)$ of $h_{t;k}(\cdot,k)$ is determined. Let

$$A := \begin{pmatrix} g_{0,t;k}(x_1,k) & \ldots & g_{u-1,t;k}(x_1,k) \\ \vdots & \ddots & \vdots \\ g_{0,t;k}(x_{N_1},k) & \ldots & g_{u-1,t;k}(x_{N_1},k) \end{pmatrix}. \tag{6}$$

If A^TA is regular (usual case) the normal equation $A^TA\mathbf{b} = A^T\mathbf{i}_t$ has unique solution

$$\widetilde{\mathbf{b}}^* = (A^TA)^{-1}A^T\mathbf{i}_t, \qquad \text{with} \quad \widetilde{\mathbf{b}}^* := (\widetilde{\beta}_0^*, ..., \widetilde{\beta}_{u-1}^*), \quad \text{and} \tag{7}$$

$$\widetilde{h}_{t;k}^*(\cdot,k) = \sum_{j=0}^{u-1} \widetilde{\beta}_{j,t;k}^* g_{j,t;k}(\cdot,k) \quad \text{(least square estimate of } h_{t;k}^*(\cdot,k)\text{)}. \tag{8}$$

The coefficients $\tilde{\beta}_{0,t;k}^*, \ldots, \tilde{\beta}_{u-1,t;k}^*$ are called β-characteristic.

In the second profiling step the covariance matrix C of the noise vector $\mathbf{R_t}$ is estimated, finally yielding a density for the random vector $\mathbf{I_t}(x,k)$. From an information theoretic point of view it seems to be advisable to consider as many time instants $t_1 < \cdots < t_m$ as possible. Unfortunately, then the covariance matrix C is often 'almost' singular so that even moderate estimation errors in $\widetilde{C}$ may amplify drastically in $\widetilde{C}^{-1}$ (needed to calculate $f_C(\cdot)$), and matrix inversion becomes an ill-posed numerical problem. Since C and its estimate $\widetilde{C}$ are symmetric positive semi-definite matrices an orthogonal matrix $P \in O(m)$ exists, for which $P^T\widetilde{C}P = \widetilde{D}_m$ with $\widetilde{D}_m = \mathrm{diag_m}(\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_\mathrm{m})$. The diagonal elements $\widetilde{\lambda}_1 \geq \cdots \geq \widetilde{\lambda}_m \geq 0$ (eigenvalues of $\widetilde{C}$), and the j^{th} column v_j of P is an eigenvector of $\widetilde{C}$ to eigenvalue $\widetilde{\lambda}_j$ (main axis transformation). If the first s eigenvalues are considerably larger than the others, i.e. $\widetilde{\lambda}_{s+1} \ll \widetilde{\lambda}_s$, we concentrate on that subspace of $\mathbb{R}^\mathrm{m}$, which is spanned by the eigenvectors $v_1, \ldots, v_s$. More precisely, if P_s denotes the $(m \times s)$-matrix with columns $v_1, \ldots, v_s$ then

$$P_s^T\widetilde{C}P_s = \widetilde{D}_s \quad \text{with } \widetilde{D}_s = \mathrm{diag_s}(\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_\mathrm{s}) \quad \text{(PCA)}. \tag{9}$$

If the random vector Y is $\mathcal{N}_m(0,C)$-distributed then P_s^TY is $\mathcal{N}_s(0, P_s^TCP_s)$-distributed ([9]), i.e. has the s-dimensional normal density f_{D_s}. For large m it is not advisable to calculate P_s and $\widetilde{D}_s$ via main axis transformation of $\widetilde{C}$. Instead, one should apply the singular value decomposition [9] as it is numerically more stable.

Attack Phase. In the attack phase the adversary performs N_3 measurements at the target device and obtains power vectors $\mathbf{i_t}(x_1, k^\dagger), \ldots, \mathbf{i_t}(x_{N_3}, k^\dagger)$ with the unknown subkey $k^\dagger$. The adversary decides for that subkey candidate $k^* \in \{0,1\}^s$ that maximizes

$$\prod_{l=1}^{N_3} f_{\widetilde{D}_s}\left(P_s^T\left(\mathbf{i_t}(x_l,k^\dagger) - \widetilde{\mathbf{h}}_{\mathbf{t};k}^*(x_l,k^*)\right)\right). \tag{10}$$

4.2 SA-OTM: A New Variant of SA

In the following we assume that the power traces are labelled in the same order as they have been recorded, and that the data-independent offset drifts slowly. We denote the offsets at time t by $\tau_{t;1}, \tau_{t;2}, \ldots$ where the second index indicates

the number of the power trace. In particular, $\tau_{t;l} - \tau_{t;l+1} \approx 0$ for all $l \geq 1$. For 'normal' SA $\widetilde{\beta}^*_{0,t,k}$ estimates the average power consumption in the profiling phase. Note that this average might differ from the corresponding value within the attack phase. Moreover, regarding the measurements of the DPA contest v2 the ratio $|\widetilde{\beta}^*_{0,t,k}| / \sum_{j=1}^{8} |\widetilde{\beta}^*_{j,t,k}| \approx 70$, and hence even moderate relative differences in $\widetilde{\beta}^*_{0,t,k}$ might have considerable impact on the attack efficiency. We refine (5) and get

$$I_t(x_l, k) = h_t(x_l, k) + \tau_{t;l} + R_t \,. \tag{11}$$

In particular, $I_t(x_l, k) \sim N(h_t(x_l, k) + \tau_{t;l}, \sigma^2)$. Of course, if $\tau_{t;l} = 0$ for all power traces (11) reduces to (5). SA-OTM applies to the enhanced mathematical model (11).

SA-OTM: Profiling Phase

Estimation of $h^{*\circ}_{t,k}$ and of the β-characteristic Since the drifting offset $\tau_{t;l}$ only affects the coefficient $\beta_{0,t;k}$ in contrast to 'normal' SA we do not aim at $h^*_{t,k}(\cdot, k)$ but at $h^{*\circ}_{t,k}(\cdot, k) := \sum_{j=1}^{u-1} \beta^*_{j,t;k} g_{j,t;k}(\cdot, k)$. In place of $\mathcal{F}_{u,t;k}$ we consider the subspace

$$\mathcal{F}^{\circ}_{u,t;k} := \{h' \colon \{0,1\}^p \times \{k\} \to \mathbb{R} \mid \mathrm{h}' = \sum_{\mathrm{j}=1}^{\mathrm{u}-1} \beta'_{\mathrm{j,t,k}} \mathrm{g_{j;t;k}} \text{ with } \beta'_{\mathrm{j,t,k}} \in \mathbb{R}\}, \tag{12}$$

i.e., we neglect the first basis vector $g_{0,t;k} = 1$. The straight-forward approach is to proceed as in 'normal' SA, by simply cancelling the first column of matrix A (Eq. (6)).

Alternatively, one may consider differences of consecutive power measurements. More precisely, for $l = 1, \ldots, N_1 - 1$ let $d_{j,t,k}(x_l, x_{l+1}, k) := g_{j,t;k}(x_l, k) - g_{j,t;k}(x_{l+1}, k)$ and $di_t(x_l, x_{l+1}, k) := i_t(x_l, k) - i_t(x_{l+1}, k)$. Further, we define the $(N_1 - 1)$-dimensional vector $\mathbf{\Delta i}_t := (di_t(x_l, x_2, k), \ldots, di_t(x_{N_1-1}, x_{N_1}, k))$ and

$$A^{\circ} := \begin{pmatrix} d_{1,t;k}(x_1, x_2, k) & \ldots & d_{u-1,t;k}(x_1, x_2, k) \\ \vdots & \ddots & \vdots \\ d_{1,t;k}(x_{N_1-1}, x_{N_1}, k) & \ldots & d_{u-1,t;k}(x_{N_1-1}, x_{N_1}, k) \end{pmatrix}. \tag{13}$$

If the $(u - 1 \times u - 1)$ dimensional matrix product $(A^{\circ T} A^{\circ})$ is regular then in analogy to (7) and (8) we obtain

$$\widetilde{\mathbf{b}}^{*\circ} = (A^{\circ T} A^{\circ})^{-1} A^{\circ T} \mathbf{\Delta i}_t \qquad \text{with} \quad \widetilde{\mathbf{b}}^{*\circ} := (\widetilde{\beta}^*_1, ..., \widetilde{\beta}^*_{u-1}), \text{ and} \tag{14}$$

$$\widetilde{h}^{*\circ}_{t;k}(\cdot, k) = \sum_{j=1}^{u-1} \widetilde{\beta}^*_{j,t,k} g_{j,t;k}(\cdot, k) \quad \text{(least square estimate of } h^{*\circ}_{t;k}(\cdot, k)) \,. \tag{15}$$

For infinite sample size N_1 the estimates $\widetilde{\beta}^*_{1,t;k}, \ldots, \widetilde{\beta}^*_{u-1,t;k}$ from both estimation methods ('straight-forward', 'difference method') converge to the exact parameter values $\beta^*_{1,t;k}, \ldots, \beta^*_{u-1,t;k}$. For the power traces from the DPA contest v2 the

difference method turned out to be more efficient (higher rate of convergence), which should be due to the fact that $\beta^*_{0,t;k}$ clearly dominates the other coefficients. Note that in the first profiling step it is a (reasonable) option to use differences of power traces while it is unavoidable in the second profiling step and in the attack phase.

Estimation of the Distribution of $\mathbf{R_t}$ *and PCA.* Since the offsets $\tau_{t;l}$ are unknown, we apply OTM. In fact, since $\tau_{t;l} - \tau_{t;l+1} \approx 0$ and

$$\begin{aligned}&(\mathbf{I_t}(x_l,k) - \mathbf{I_t}(x_{l+1},k)) - (\mathbf{h}^{*\circ}_{\mathbf{t}}(x_l,k) - \mathbf{h}^{*\circ}_{\mathbf{t}}(x_{l+1},k)) \approx \\ &\mathbf{I_t}(x_l,k) - \mathbf{h_t}(x_l,k) - \tau_{t;l} - (\mathbf{I_t}(x_{l+1},k) - \mathbf{h_t}(x_{l+1},k) - \tau_{t;l+1}) \sim \mathcal{N}(0,2C).\end{aligned} \tag{16}$$

Consequently, we go for an estimate of $2C$ instead of C. Now let $\mathbf{w}_{\mathbf{t},l;k} := \mathbf{i_t}(x_l,k) - \widetilde{\mathbf{h}}^{*\circ}_{\mathbf{t}}(x_l,k)$. Then

$$\begin{aligned}\widetilde{2C} &:= \frac{1}{N_2-1}\widetilde{M}^{\circ T}\widetilde{M}^{\circ} \quad \text{with the } (m \times (N_2-1))\text{-matrix} \\ \widetilde{M}^{\circ T} &:= (\mathbf{w}_{\mathbf{t},1;k} - \mathbf{w}_{\mathbf{t},2;k}, \ldots, \mathbf{w}_{\mathbf{t},N_2-1;k} - \mathbf{w}_{\mathbf{t},N_2;k})\end{aligned} \tag{17}$$

provides an estimate for $2C$. We point out that the columns of M° are not independent. However, let M°_{ev} and M°_{odd} denote the submatrices of M°, which consist of the columns with even indices or of the columns with odd indices, respectively. For odd N_2

$$\frac{1}{N_2-1}\widetilde{M}^{\circ T}\widetilde{M}^{\circ} = \frac{1}{2}\left(\frac{2}{N_2-1}\widetilde{M}^{\circ T}_{\text{ev}}\widetilde{M}^{\circ}_{\text{ev}} + \frac{2}{N_2-1}\widetilde{M}^{\circ T}_{\text{odd}}\widetilde{M}^{\circ}_{\text{odd}}\right). \tag{18}$$

Both submatrices have independent columns, which yield estimates for $2C$ (analogously to the SA case). We point out that $\widetilde{M}^\circ_{\text{ev}}$ and $\widetilde{M}^\circ_{\text{odd}}$ are only weakly correlated since the l^{th} row of $\widetilde{M}^\circ$ is only correlated to rows $(l-1)$ and l. The matrices C and $2C$ have the same eigenspaces and thus the same transformation matrix P_s (cf. Eq. (9)). Applying the singular value decomposition to $\widetilde{M}^\circ$ yields P_s as well as estimates $\widetilde{2D_s} = 2\widetilde{D}_s$ and $\widetilde{D}_s$ for $2D_s$ and D_s, respectively.

SA-OTM: Attack Phase. We assume that the attacker has recorded N_3 measurement vectors $\mathbf{i_t}(x_1,k^\dagger),\ldots,\mathbf{i_t}(x_{N_3},k^\dagger)$ from a target device with a secret (unknown) subkey $k^\dagger$. As for SA the attacker applies a maximum likelihood estimation rule but for SA-OTM the situation becomes more complicated (Theorem 1).

Notation 2. If $F_1,\ldots,F_r$ are matrices with the same number of columns then $RV(F_1,\ldots,F_r)$ denotes the block matrix whose first rows are given by F_1, the next rows by F_2 etc.

Theorem 1. *For $l = 1, \ldots, N_3$ let $\mathbf{W}_{\mathbf{t},l;k} := \mathbf{I}_{\mathbf{t}}(x_l, k) - \mathbf{h}^{*\circ}_{\mathbf{t},k}(x_l, k)$. Then the $s(N_3-1)$-dimensional random vector*

$$\begin{aligned}\mathbf{W}_{\mathbf{t};k} &:= RV\big(P_s^T(\mathbf{W}_{\mathbf{t},1;k} - \mathbf{W}_{\mathbf{t},2;k}), P_s^T(\mathbf{W}_{\mathbf{t},2;k} - \mathbf{W}_{\mathbf{t},3;k}), \ldots, P_s^T(\mathbf{W}_{\mathbf{t},N_3-1;k} - \mathbf{W}_{\mathbf{t},N_3;k})\big)\\ &\sim N(RV(P_s^T(\tau_{\mathbf{t};1} - \tau_{\mathbf{t};2}), P_s^T(\tau_{\mathbf{t};2} - \tau_{\mathbf{t};3}), \ldots, P_s^T(\tau_{\mathbf{t};N_3-1} - \tau_{\mathbf{t};N_3})), G(D_s))\\ &\approx N(0, G(D_s)) \qquad (19)\end{aligned}$$

with the $(s(N_3-1) \times s(N_3-1))$-dimensional block tridiagonal matrix

$$G(D_s) = \begin{pmatrix} 2D_s & -D_s & & & \\ -D_s & 2D_s & -D_s & & \\ & -D_s & 2D_s & -D_s & \\ & & \ddots & \ddots & \ddots \\ & 0 & & -D_s & 2D_s & -D_s \\ & & & & -D_s & 2D_s \end{pmatrix}.$$

Proof. Since the random vectors $\mathbf{W}_{t,1;k}, \ldots, \mathbf{W}_{t,N_3;k}$ are independent the random vector

$$\mathbf{V} := RV(P_s^T(\mathbf{W}_{t,1;k}), P_s^T(\mathbf{W}_{t,2;k}), \ldots, P_s^T(\mathbf{W}_{t,N_3;k}))$$

is $\mathcal{N}\left(RV(P_s^T(\tau_{t;1}), P_s^T(\tau_{t;2}), \ldots, P_s^T(\tau_{t;N_3})), \hat{D}\right)$-distributed where $\hat{D}$ stands for the $(sN_3 \times sN_3)$-dimensional block diagonal matrix whose N_3 diagonal blocks equal D_s. We conclude $\mathbf{W}_{t;k} = L(V)$ where $L\colon \mathbb{R}^{sN_3} \to \mathbb{R}^{s(N_3-1)}$ denotes the linear mapping

$$L(RV(z_1, \ldots, z_{N_3})) := RV(z_1 - z_2, z_2 - z_3, \ldots, z_{N_3-1} - z_{N_3}).$$

By [9] (3.31) we have

$$L(\mathbf{V}) \sim \mathcal{N}(RV(P_s^T(\tau_{\mathbf{t};j} - \tau_{\mathbf{t};2}), P_s^T(\tau_{\mathbf{t};2} - \tau_{\mathbf{t};3}), \ldots, P_s^T(\tau_{\mathbf{t};N_3-1} - \tau_{\mathbf{t};N_3}), L\hat{D}L^T).$$

A careful computation verifies $L\hat{D}L^T = G(D_s)$, which proves the first assertion of Theorem 1. Since L is linear the second assertion follows from the assumption that the differences $\tau_{\mathbf{t};1} - \tau_{\mathbf{t};2}, \tau_{\mathbf{t};2} - \tau_{\mathbf{t};3}, \ldots, \tau_{\mathbf{t};N_3-1} - \tau_{\mathbf{t};N_3} \approx 0$. □

If the vector space $\mathcal{F}^{\circ}_{u,t;k}$ catches the relevant parts of the leakage then $\mathbf{h}^{*\circ}_{\mathbf{t};k}(x_l, k) - \mathbf{h}^{*\circ}_{\mathbf{t};k}(x_{l+1}, k) \approx \mathbf{h}^{*}_{\mathbf{t};k}(x_l, k) - \tau_{\mathbf{t};l} - \mathbf{h}^{*}_{\mathbf{t};k}(x_{l+1}, k) + \tau_{\mathbf{t};l+1}$, which motivates the following maximum likelihood decision rule. The adversary decides for the subkey $k \in \{0,1\}^s$, which maximizes $f_{G(\widetilde{D}_s)}$, or equivalently minimizes

$$({\mathbf{w}'_{\mathbf{t};k}}^T G(\widetilde{D}_s)^{-1} \mathbf{w}'_{\mathbf{t};k}) \text{ with} \qquad (20)$$

$$\mathbf{w}'_{\mathbf{t};k} := RV\Big(P_s^T(\mathbf{w}'_{\mathbf{t},1;k} - \mathbf{w}'_{\mathbf{t},2;k}), P_s^T(\mathbf{w}'_{\mathbf{t},2;k} - \mathbf{w}'_{\mathbf{t},3;k}), \ldots, P_s^T(\mathbf{w}'_{\mathbf{t},N_3-1;k} - \mathbf{w}'_{\mathbf{t},N_3;k})\Big)$$

and $\mathbf{w}'_{\mathbf{t},l;k} := \mathbf{i}_{\mathbf{t}}(x_l, k^{\dagger}) - \widetilde{\mathbf{h}}^{*\circ}_{\mathbf{t},k}(x_l, k)$ while $G(\widetilde{D}_s)$ is the estimate of $G(D_s)$.

Remark 1. Eq. (20) can be evaluated without inverting $G(\widetilde{D}_s)^{-1}$. Instead, one first solves the matrix-vector equation $G(\widetilde{D}_s)\mathbf{v} = \mathbf{w}'_{\mathbf{t};k}$ first, for which efficient numerical algorithms exist (e.g., iterative Krylov Methods [20]). Finally, one computes ${\mathbf{w}'}^{T}_{\mathbf{t};k}\mathbf{v}$. We point out that also these calculations could be saved by cancelling every second component in $\mathbf{w}'_{\mathbf{t};k}$ (at cost of doubling the number of attack traces!). As a compromise between efficiency and computational workload one might cancel every α^{th} component of $\mathbf{w}'_{\mathbf{t};k}$, which results a block diagonal matrix with $\frac{N_3}{\alpha}$ matrices G_l for $1 \leq l \leq \frac{N_3}{\alpha}$ and $\dim(G_l) \leq s \cdot (\alpha - 1) \ll \dim G(\widetilde{D}_s)$ in its diagonal. Here one 'wastes' $\frac{N_3}{\alpha}$ power traces for the sake of faster calculation. This method is of particular interest in context of the DPA contest v2 since the contest rules demand the continued evaluation of an attack for increasing sets of power traces. In our experiments (Sect. 6) we used $\alpha = 200$, without claiming that the choice of α is optimal.

5 On the Selection of Stochastic Leakage Models

The approximator of the leakage function $h_{t;k}$ (i.e., $h^*_{t;k}$ or $h^{*\circ}_{t;k}$) is close only for an appropriate subspace $\mathcal{F}_{u,t;k}$. The appropriateness depends on the leakage model and thus on the concrete subspace. In this section we consider different subspaces that may be used for attacks on the last round of an AES-128 hardware implementation. In [10] a 9-dimensional subspace $\mathcal{F}_{9,t;k}$ was investigated in detail (SA). The selection of $\mathcal{F}_{9,t;k}$ is reasonable if (one assumes that) the side-channel leakage is only caused by the sum of the individual transitions on all bit lines. High-dimensional subspaces also capture effects that arise from interactions between the transitions on two or more bit lines. Such effects occur due to properties of internal circuit structures, e.g., propagation glitches or cross-talk phenomena during the metastable phase of the registers, which is a well-known problem in CMOS VLSI Circuit Design [12,5]. In Sect. 6 we consider $\mathcal{F}^{\circ}_{u,t;k}$ for $u \in \{9, 37, 93, 163, 219, 247, 255, 256\}$. Recall that $\dim \mathcal{F}^{\circ}_{u,t;k} = u - 1$. Of course, also high-dimensional subspaces keep the regression linear.

The possibility of applying high-dimensional subspaces was already pointed out in [17,16]. In [13], Renauld et al. analyzed the information theoretic impact of high-dimensional leakage models on the mutual information. To the best of the authors' knowledge very high-dimensional subspaces have not been evaluated in concrete attacks yet.

5.1 High-Dimensional Subspaces for SA-OTM

With regard to an ordinary hamming distance model we first consider the 8-dimensional subspace $\mathcal{F}^{\circ}_{9,t;k}$ which exploits the corresponding intermediate value of the 9^{th} round XORed with the 10^{th} round key. More precisely, we select the following basis vectors

$$g_{j,t;k_{(y)}}((x_{(z)},x_{(y)}),k_{(y)}) = (\underbrace{(x_{(z)} \oplus S^{-1}(x_{(y)} \oplus k_{(y)}))_j}_{:=(\phi(x_{(z)},x_{(y)},k_{(y)}))_j := \hat{g}_{j,t;k}} - 2^{-1}) \qquad (21)$$

for $j = 1, \ldots, 8$.

The subtrahend 2^{-1} ensures $E_X(g_{j,t,k_{(y)}}(X, k_{(y)})) = 0$ for independent and uniformly distributed random variables $X_{(y)}$ and $X_{(z)}$, a reasonable model for two ciphertext bytes. The indices y and z are chosen according to the distance model of the AES design (cf. [10,8]).

Moreover, we consider high-dimensional subspaces. To simplify we introduce new notation. First, $\mathcal{B}_1 := \{g_{1,t;k_{(y)}}, \ldots, g_{8,t;k_{(y)}}\}$ collects all basis vectors from Eq. (21), which capture the contribution of the individual bit lines. Moreover, for $2 \leq i \leq 8$ the set

$$\mathcal{B}_i := \{\hat{g}_{j_1,t;k_{(y)}} \cdots \hat{g}_{j_i,t;k_{(y)}} - 2^{-i} \mid 1 \leq j_1 < \ldots < j_i \leq 8\} \qquad (22)$$

contains all unordered i-fold products of elements in $\mathcal{B}_1$ minus 2^{-i}. (A typical element in $\mathcal{B}_2$ is $\hat{g}_{4,t;k_{(y)}} \cdot \hat{g}_{7,t;k_{(y)}} - 2^{-2}$.) The subtrahend 2^{-i} ensures the zero-mean property for all elements of $\mathcal{B}_i$. Table 1 provides the basis vectors for all relevant subspaces (the elements of the sets in the second column).

Table 1. Set of basis functions for each subspace

$\dim(\mathcal{F}^{\circ}_{u,t;k})$ $(= u-1)$	Set of basis functions
8	$\mathcal{B}_1$
36	$\mathcal{B}_1 \cup \mathcal{B}_2$
92	$\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3$
162	$\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4$
218	$\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathcal{B}_5$
246	$\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathcal{B}_5 \cup \mathcal{B}_6$
254	$\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathcal{B}_5 \cup \mathcal{B}_6 \cup \mathcal{B}_7$
255	$\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathcal{B}_5 \cup \mathcal{B}_6 \cup \mathcal{B}_7 \cup \mathcal{B}_8$

5.2 Leakage Models for the Stochastic Approach

As pointed out in Subsect. 4.2 the subspaces for SA-OTM are similar to the subspaces for SA, just the first basis vector $g_{0,t;k_{(y)}}$ is omitted. In particular, the subspace $\mathcal{F}_{9,t;k}$ is spanned by

$$g_{0,t;k_{(y)}}((x_{(z)},x_{(y)}),k_{(y)}) = 1 \qquad (23)$$
$$g_{j,t;k_{(y)}}((x_{(z)},x_{(y)}),k_{(y)}) = ((x_{(z)} \oplus S^{-1}(x_{(y)} \oplus k_{(y)}))_j - 2^{-1}) \quad \text{for } j = 1, \ldots, 8\,.$$

We define $\mathcal{B}_0 := \{g_{0,t;k_{(y)}}\}$, and the construction of high-dimensional subspaces follows analogously to Tab. 1 with additional basis vector $\mathcal{B}_0$.

5.3 Symmetry

References [10,16] consider leakage models with symmetries (for SA). In fact, the basis vectors $g_{j;t;k_{(y)}}$ from Subsect. 5.1 and 5.2 can be expressed by a composition of a key-independent function $\overline{g}_{j,t}\colon \{0,1\}^8 \to \mathbb{R}$ with the mapping $\phi\colon \{0,1\}^8 \times \{0,1\}^8 \times \{0,1\}^8 \to \{0,1\}^8$, given by

$$\phi(x_{(z)}, x_{(y)}, k_{(y)}) := (x_{(z)} \oplus S^{-1}(x_{(y)} \oplus k_{(y)}))\,.$$

This essentially reduces the argument of $g_{j;t;k_{(y)}}$ from 24 to 8 bits. If the leakage function $h_{t,k_{(y)}}((x_{(z)}, x_{(y)}), k_{(y)})$ also depends on its arguments only through $\phi(x_{(z)}, x_{(y)}, k_{(y)})$ one can compute $h^*_{t,k'_{(y)}}((\cdot,\cdot), k'_{(y)})$ for each $k'_{(y)}$ if $h^*_{t,k_{(y)}}((\cdot,\cdot), k_{(y)})$ is known for arbitrary subkey $k_{(y)}$. In particular, for uniformly distributed $(X_{(y)}, X_{(z)})$ for each $j < u$

$$\beta_{j,t;k'_{(y)}} \equiv \beta_{j,t} \quad \text{for all } k'_{(y)} \in \{0,1\}^8. \tag{24}$$

Reference [8] explains how to verify, resp. to falsify, whether symmetry assumptions are indeed valid, and a symmetry metric is introduced. Eq. (24) says that the coefficients $\beta_{j,t,\cdot}$ are identical for all admissible subkeys. This property allows to use all 1000.000 power traces of the template base (though belonging to different (sub)keys) jointly in a single least square estimation process. This gives more stable results, and (for each key byte) profiling step 1 has to be carried out only once.

6 Experimental Analysis

In this section we compare the efficiency of SA-OTM and of SA on basis of the DPA contest v2 power traces. We apply the leakage models from Sect. 5. The DPA contest v2 provides two data bases: A template base with 1.000.000 power traces (to develop an attack), and a public base, which contains power traces for 32 fixed keys, 20.000 traces for each key (to test the attack). The organizers of the contest evaluated the submitted attacks on a (non-public) private base to avoid "biased" attacks. The measurements were recorded on the SASEBO-G II FPGA-evaluation board [14] using a Virtex 5 FPGA [19]. Each encryption (AES with 128 bit keys) takes 10 clock cycles, and the SBOX realization is based on a composite field [15]. In analogy to the DPA contest v2 we calculate the partial success rate (PSR) and the global success rate (GSR) to compare the efficiency of the particular attacks. The PSR is the probability that the correct subkey is ranked first among all possible subkeys, while GSR denotes the probability that the complete key is ranked first. We are mainly interested in the minimum number of power traces for which the PSR is stable above 80% (i.e. the 'worst' byte is stable in $> 80\%$ of the experiments), and in the minimum number of power traces for which the GSR is stable above 80% ($\rightarrow$ evaluation criteria for the DPA contest v2).

As already mentioned the template base consists of 1.000.000 traces, i.e. for each subkey $\approx$ 4.000 traces. This number is too small for a sufficiently precise estimation of the β-characteristic for each key. However, due to the symmetry properties of the attacked implementation (cf. Subsect. 5.3) we could circumvent this problem. Accordingly, we computed the coefficients $\widetilde{\beta}^*_{j,t}$ (cf. Eq. (24)) on basis of all 1.000.000 power traces.

The application of PCA to the Covariance matrices $\widetilde{C}$ and $2\widetilde{C}$ showed that the first eigenvalue $\widetilde{\lambda}_1$ is at least 20 times larger than the other eigenvalues $\widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_m$. Consequently, we selected $s = 1$, and hence P_s is an $(m \times 1)$ matrix (cf. Eq. (9)). For the evaluation of the PSR and the GSR we used the power

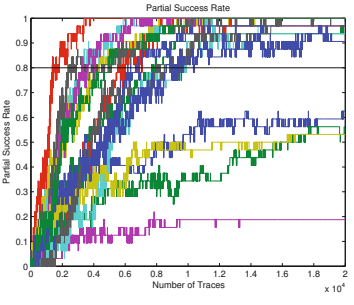


Fig. 4. Partial success rate: SA with a 9-dim. model

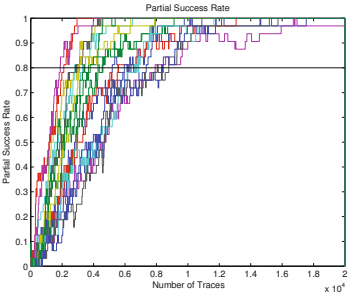


Fig. 5. Partial success rate: SA-OTM with a 8-dim. model

traces from the public base. Figure 4 depicts the PSR for the SA with the 9-dimensional leakage model from Eq. (23). Each curve corresponds to one of the 16 subkeys. Figure 5 shows the PSR for SA-OTM with the 8-dimensional leakage model (e.g., Eq. (21)). All bytes achieve the 80% threshold, and except for one subkey, even the 100% threshold. Figure 6 and 7 depict the PSR for SA and

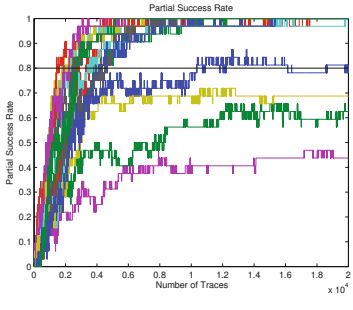


Fig. 6. Partial success rate: SA with a 37-dim. model

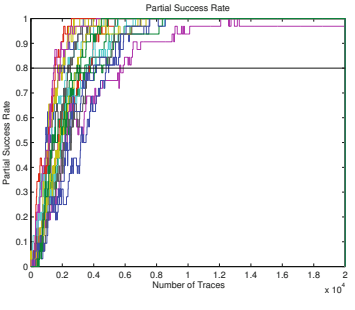


Fig. 7. Partial success rate: SA-OTM with a 36-dim. model

SA-OTM using the 37-dimensional and 36-dimensional model, which capture the individual leakage of each bit line *and* the leakage caused by the interaction between two arbitrary bit lines. Evidently, these leakage models describe the existing leakage more precisely. However, for SA the PSR criterion fails due to the same 4 bytes. Compared to the 8-dimensional leakage model for SA-OTM the minimum number of traces with stable PSR $> 80\%$ drops down from 8781 to 5876. The 93- and the 92-dimensional leakage model additionally capture the

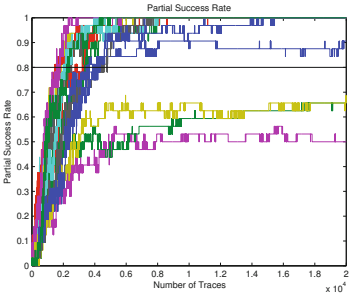


Fig. 8. Partial success rate: SA with a 93-dim. model

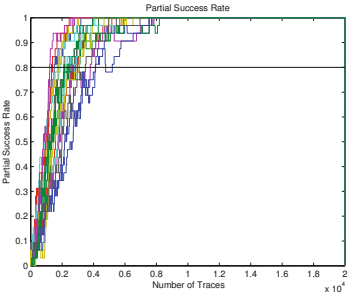


Fig. 9. Partial success rate: SA-OTM with a 92-dim. model

leakage that arises from the combinations of three bit lines. Experimental results are depicted in Figure 8 and Figure 9. The 93-dimensional model (SA) improves the PSR, but 3 bytes still do not reach 80% PSR. SA-OTM reaches the PSR stable above 80% after 5195 traces. The significant improvement of the PSR for specific bytes does not necessarily imply that the drifting offset only influences those bytes. It rather underlines that not all subkey bytes 'leak' in the same way. One might conjecture that those subkey bytes, which have less influence on the overall leakage are more affected by the drifting offset than the others. Figure 10 shows the GSR. SA-OTM requires about 6734 traces to achieve a GSR stable $> 80\%$. A GSR of 100% is only archived for SA-OTM with the 92-dimensional leakage model.

The best attack that was submitted during the contest achieves a PSR stable above 80% for 5.890 traces and a stable GSR $> 80\%$ for 7.061 traces. SA-OTM with the 92-dimensional model outperforms these benchmarks. Moreover, we computed the success rates of SA-OTM for the 162-, 218, 246-, 254-, 255- dimensional model, which increased this success rate further. These results indicate that the 218-dimensional subspace $\mathcal{F}^{\circ}_{219,t;k}$, which is spanned by the basis vectors in $\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_3 \cup \mathcal{B}_4 \cup \mathcal{B}_5$, seems to essentially capture the leakage. Tab. 2 contains all results for the public base and the private base (as far as known). For the private base a third contest criterion, the maximal partial guessing entropy below 10 (max PGE below 10) [18], is listed.

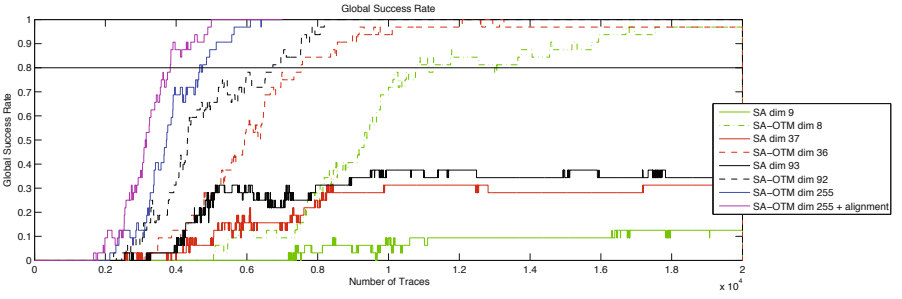


Fig. 10. Global success rate of SA and SA-OTM for different models

Remark 2. Following the expositions in Sect. 2 one might simply try to combine 'normal' SA with vertical trace alignment. However, as explained below also in combination with vertical alignment SA-OTM remains more efficient than SA.

Vertical Trace Alignment. We combined SA and SA-OTM with vertical trace alignment, a well-known method in power analysis. We 'normalized' each measurement trace to mean zero (over the whole trace), i.e. we computed *aligned* $i_{\mathbf{t}}(x_l, k) = i_{\mathbf{t}}(x_l, k) - \text{mean}(i_{\mathbf{t}}(x_l, k))$, with $\mathbf{t}$ ranging over the complete trace. However, even then SA with the 93-dimensional model did not exceed PSR $> 80\%$ for all bytes.

Since SA-OTM itself solves the problem with the drifting offset our goal here was to reduce the impact of outliers. Apart from the drifting offset, Figure 1 displays a few extreme values that might be caused by such outliers, which result from measurement errors or any other interference during the acquisition.

Table 2. Success rate (PSR stable $> 80\%$ / GSR stable $> 80\%$), and max PGE below 10 for the private base

Attack	Public Base	Private Base[1]
SA-OTM: dim 8	(8781 / 13020)	unknown
SA-OTM: dim 36	(5876 / 7533)	unknown
SA-OTM: dim 92	(5195 / 6734)	(4358 / 5571), 1.894
SA-OTM: dim 162	(4353 / 6144)	unknown
SA-OTM: dim 218	(3552 / 4564)	unknown
SA-OTM: dim 246	(3769 / 4691)	unknown
SA-OTM: dim 254	(3720 / 4740)	unknown
SA-OTM: dim 255	(3718 / 4748)	unknown
SA-OTM: dim 255 incl. alignment	(2682 / 3836)	(2748 / 3589), 1.356
Best submitted attack during the first & second period	unknown	(5890 / 7061), 2.767

[1] See http://www.dpacontest.org/v2/hall_of_fame.php for the results on the private base.

Alternatively, one could also try to identify and omit the outliers. SA-OTM with the 255-dimensional leakage model and vertical alignment SA-OTM achieves a PSR stable $> 80\%$ within 2748 traces and a GSR stable above 80% within 3589 traces (private base). These results reduce the required number of traces to 50% compared to the best submitted attack during the contest, cf. Tab. 2.

Further Work / Open Problems Our analysis raises several questions. Can the drifting offset be effectively prevented in practice? What is the smallest subspace that captures all relevant parts of the compromising leakage? Do different types of implementations demand different subspaces? Another ambitious topic for future work could be an automatized search for optimal (high-dimensional) subspaces, which finally might yield to appropriate basis vector selection methods.

7 Conclusion

In this contribution we investigated two fundamental problems that may affect the efficiency of profiling based attacks, and we developed efficient solutions. Drifting offsets (caused by temperature variations) cause difficulties for attacks, which consider (implicitly or explicitly) the average power consumption (typically profiling based attacks). We introduced a new method, denoted as the Offset Tolerant Method (OTM), which considers differences of consecutive pairs of power traces. We adjusted OTM to the stochastic approach (SA), abbreviated by SA-OTM. In presence of a drifting offset SA-OTM turned out to be clearly more efficient than SA, even in combination with vertical trace alignment.

We further addressed the problem of how to select suitable leakage models, which shall represent the compromising leakage as precise as possible. Our results show that leakage may also arise from the interaction of several bit lines. This effect can only be captured by high-dimensional leakage models. Combining these two improvements we achieved the best results of all participants of the DPA contest v2. Further research work might consider open problems formulated at the end of Sect. 6 or concentrate on improvements of SA-OTM, maybe in combination with alternative dimension reduction techniques.

Acknowledgment. The work presented in this contribution was supported by the German Federal Ministry of Education and Research (BMBF) in the project *Resist* through grant number 01IS10027A. We thank Christian Brandt for the heating cabinet framework.

References

1. Archambeau, C., Peeters, E., Standaert, F.X., Quisquater, J.J.: Template Attacks in Principal Subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)

2. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
3. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, C., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
4. DPA contest v2, http://www.dpacontest.org/
5. Eo, Y., Eisenstadt, W., Jeong, J.Y., Kwon, O.K.: A new on-chip interconnect crosstalk model and experimental verification for CMOS VLSI circuit design. IEEE Transactions on Electron Devices 47(1), 129–140 (2000)
6. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
7. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006)
8. Heuser, A., Kasper, M., Schinder, W., Stöttinger, M.: How a Symmetry Metric Assists Side-Channel Evaluation - A Novel Model Verification Method for Power Analysis. In: 14th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2011). IEEE (2011)
9. Kardaun, O.: Classical Methods of Statistics. Springer, Heidelberg (2005)
10. Kasper, M., Schindler, W., Stöttinger, M.: A Stochastic Method for Security Evaluation of Cryptographic FPGA Implementations. In: FPT 2010, pp. 146–154. IEEE Press (2010)
11. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
12. Nieuwland, A.K., Katoch, A., Meijer, M.: Reducing Cross-Talk Induced Power Consumption and Delay. In: Macii, E., Koufopavlou, O.G., Paliouras, V. (eds.) PATMOS 2004. LNCS, vol. 3254, pp. 179–188. Springer, Heidelberg (2004)
13. Renauld, M., Standaert, F.X., Veyrat-Charvillion, N., Kamel, D., Flandre, D.: A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 109–128. Springer, Heidelberg (2011)
14. SASEBO GII, http://www.rcis.aist.go.jp/special/SASEBO/SASEBO-GII-en.html
15. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
16. Schindler, W.: Advanced Stochastic Methods in Side Channel Analysis on Block Ciphers in the Presence of Masking. Math. Crypt. 2, 291–310 (2008)
17. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
18. Standaert, F.X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
19. Virtex-5 FPGA User Guide (2010)
20. Vorst, H.A.V.D.: Iterative Krylov Methods for Large Linear Systems. Cambridge University Press, Cambridge (2003)

Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis

Lejla Batina[1,2], Jip Hogenboom[3,⋆], and Jasper G.J. van Woudenberg[4]

[1] Radboud University Nijmegen, ICIS/Digital Security group
Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands
lejla@cs.ru.nl
[2] K.U.Leuven ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
lejla.batina@esat.kuleuven.be
[3] KPMG Advisory N.V.
Laan van Langerhuize 1, 1186 DS Amstelveen, The Netherlands
Hogenboom.Jip@kpmg.nl
[4] Riscure BV
Delftechpark 49, 2628 XJ Delft, The Netherlands
vanwoudenberg@riscure.com

Abstract. Differential Power Analysis (DPA) is commonly used to obtain information about the secret key used in cryptographic devices. Countermeasures against DPA can cause power traces to be misaligned, which reduces the effectiveness of DPA. Principal Component Analysis (PCA) is a powerful tool, which is used in different research areas to identify trends in a data set. Principal Components are introduced to describe the relationships within the data. The largest principal components capture the data with the largest variance. These Principal Components can be used to reduce the noise in a data set or to transform the data set in terms of these components. We propose the use of Principal Component Analysis to improve the correlation for the correct key guess for DPA attacks on software DES traces and show that it can also be applied for other algorithms. We also introduce a new way of determining key candidates by calculating the absolute average value of the correlation traces after a DPA attack on a PCA-transformed trace. We conclude that Principal Component Analysis can successfully be used as a preprocessing technique to reduce the noise in a trace set and improve the correlation for the correct key guess using Differential Power Analysis attacks.

Keywords: Side-channel cryptanalysis, DPA, countermeasures, PCA.

1 Introduction

Side-channel attacks are indirect methods which are used to find secret keys in cryptographic devices. On these devices, cryptographic algorithms are

⋆ This work was done when the author was with Radboud University Nijmegen.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 383–397, 2012.

implemented to ensure encrypted communication. Smart cards can sometimes contain a software implementation of cryptographic algorithm but often they also include a cryptographic co-processor, where the larger devices usually have dedicated hardware implementations. The secret keys used for the algorithms are usually well protected within these devices.

A widely used method to recover secret keys is by using side-channel information. Side-channel attacks make use of leaked physical information, such as power consumption, electromagnetic (EM) radiation etc. This information is leaked because of weaknesses in the physical implementation of the algorithm.

An example of a widely used side-channel attack is Differential Power Analysis (DPA) [8]. The power consumption of a cryptographic device is dependent on the data being processed and in particular on the secret key that is used for encryption (decryption). This power consumption is measured while the secret key is manipulated within a cryptographic device and the corresponding power traces are collected. Performing DPA on the traces assumes doing some statistics on the power measurements and modeled traces. In this way, the attacker is using a side-channel distinguisher (e.g. DoM [8], Pearson correlation coefficient [3], Mutual Information [5] etc.) on the actual traces and the predictions for the measurements in order to test the hypothesis about (the part of) the used cryptographic key.

To defend against side-channel analysis, manufacturers of cryptographic devices usually implement countermeasures on their devices to complicate DPA substantially. Common methods include masking the sensitive values of data i.e. the variables depending on known data (e.g. plaintext) and the hypothesized key, and hiding of the dependency of power on data [9] in specific time moments. The latter can be obtained by various means e.g. random process interrupts (RPI) [4], random process order, unstable clocks etc. For example, when RPI are used as a countermeasure the position of the leakage that is exploited by DPA can shift a few clock cycles. In this way, locating the specific time points, where the key is processed is further obfuscated. Due to all those countermeasures pre-processing power traces has become an important step in side-channel analysis.

Principal Component Analysis (PCA) [7] is a technique which is widely used to reduce the noise or the dimensionality in a data set, while retaining the most variance. PCA results in a new ordered set of vectors that form an orthogonal basis for a data set. Each basis vector, or Principal Component (PC), captures the highest variance of all following PCs. PCA is used in many different domains such as gene analysis and face recognition.

An interesting property of PCA in the context of trace set analysis is that correlating samples in time are projected onto one or a few PCs. As time-domain traces often have multiple samples where leakage is presented, we hypothesize that these samples will be projected onto only a few PCs. This implies effective filtering strategies that are possible when other PCs are filtered out, and additionally CPA could be performed on PCA transformed traces. In this paper we explore these ideas. We show several directions for the PCA tools to improve side-channel analysis in pre-processing as well as in the actual analysis.

We first use PCA to transform trace sets such that, even when leakage of the key bits (through the sensitive variables) appears at different points in time, the trace set can be still analyzed with DPA. As PCA transform can reduce the dimension of the trace set, we also show how to transform the original data to a new trace set in terms of the Principal Components. After applying this transformation, the most variance within the data is included in the first part (the first Principal Components) of the transformed trace set. This fact is also used by [14], where the authors defined a new side-channel distinguisher based on the first principal component.

There were several attempts to deploy PCA in side-channel cryptanalysis but the full potentials of it are yet to be fully unleashed. First investigation was performed by Bohy at al. [2]. They considered PCA as a method to improve power attacks. However, their results cover the effects of PCA on SPA only, while further studies extend to PCA on DPA and CPA.

Archambeau et al. [1] used PCA for template attacks. In this approach the traces are first transformed by PCA in order to perform interest point selection. Indeed, in the pre-processing phase the attacker builds templates in order to complete the profiling phase by using a clone of the device under attacks. Then, the templates are used to mount an attack on the real device. The top principal components are used to capture the maximum variance between different template classes.

In contrast to [1] Souissi et al. [14] used PCA not as pre-processing tool but as a common side-channel distinguisher. The new distinguisher has the usual steps of differential power analysis (DPA [8] or CPA [3]) that consists of computational phase only and does not require an identical device for profiling.

Our Contribution. Our work is not considering the scenario of template attacks (that are assumed to be the strongest side-channel attacks [15]) nor we deploy PCA as yet another distinguisher. We introduce PCA as a suitable pre-processing technique on the power traces to enhance the results of DPA. The two benefits of PCA we observe are noise reduction and a PCA transformation (leading to more efficient DPA). Both were analyzed and several experiments are performed on unprotected and implementations with countermeasures. Additionally, we investigate the suitability of PCA on misaligned traces where our results show good results when compared to e.g. static alignment. We conclude that PCA has many potentials in the field of side-channel cryptanalysis and we expect more research to evolve.

The remainder of this paper is organized as follows. Section 2 describes some background on PCA and its applications. Our experiments with PCA related to DPA are described in Sect. 3. We compare our results to some previous work in Sect. 4. In Sect. 5 we conclude this work and discuss our findings.

2 Principal Component Analysis

Principal Component Analysis (PCA) is a multivariate technique that is widely used to reduce the noise or the dimensionality in a data set, while retaining the most variance. The origin of PCA goes back more than 100 years ago to Pearson [12] and also later a relevant formulation is due to Hotelling [6].

PCA computes a set of new orthogonal variables (by means of eigenvectors) with the decreasing variances within the data set, producing Principal Components (PC). The largest variance is captured by the highest component. PCA is used in many different domains such as gene analysis and face recognition.

When we consider PCA in terms of power measurements, we have a data set where the dimensionality is equal to the number of samples and the number of observations is equal to the number of traces. This means that the number of Principal Components which can be deduced from a power trace is (at most) equal to the number of samples.

The main drawback of PCA is that a covariance matrix of $n * n$ (where n is the number of samples) must be calculated. This means that the calculation time increases quadratically relative to the number of samples.

2.1 Example

In order to illustrate the way PCA works, we give a small example for a two-dimensional (x,y) data set with 50 observations [7]. In Fig. 1 (left) a plot of this data set is given. The first principal component is required to have the largest variance. The second component must be orthogonal to the first component while capturing the largest variance within the data set in that direction. These components are plotted in Fig. 1. This results in components which are sorted by variance, where the first component captures the largest variance. If we transform the data set using these principal components, the plot given in Fig. 1 (right)

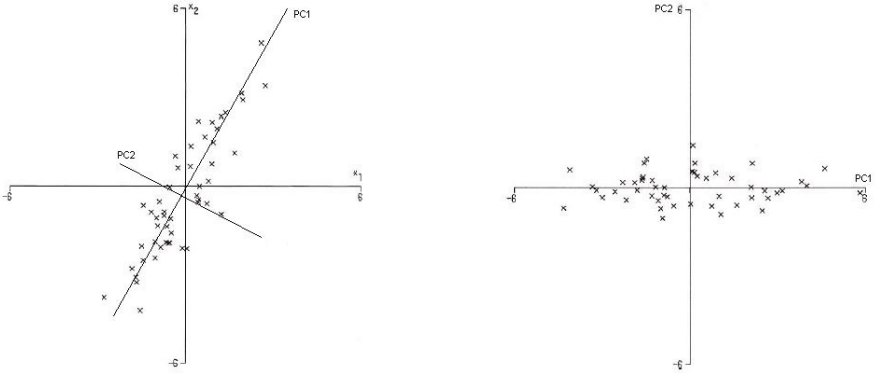


Fig. 1. A plotted data set with both of its principal components (left) and plot of the transformed data set with respect to the both principal components (right) [7]

will be obtained. This plot clearly shows that there is a larger variation in the direction of the first principal component.

2.2 PCA Transformation

In general PCA is used when trying to extract the most interesting information from data with large dimensions. More precisely, PCA attempts to find a new representation of the original set by constructing a set of orthogonal vectors spanning a subspace of the initial space. The new variables that are linear combinations of the starting ones are called principal components.

Power traces usually have large dimensions, and one would like to find the information of the key leakage within them.

In order to calculate PCA, the following few steps have to be performed [13].

- First, the mean is computed as the average over all n dimensions (samples).

$$M_n = \frac{\sum_{i=1}^{n} T_{i,n}}{n}$$

  where $T_{i,n}$ means all traces are considered as n-dimensional vectors.
  This mean M_n is afterwards subtracted from each of the dimensions n for each trace T_i.

$$T_{i,n} = T_{i,n} - M_n$$

- The covariance matrix Σ is constructed. A covariance matrix is a matrix whose (i, j)th element is the covariance between the ith and jth dimension of each trace. This matrix will be a $n * n$ matrix where n is equal to the number of samples (dimension) of the power traces. This means that the calculation time increases quadratically relative to the number of samples.
  In general, the covariance for two n-dimensional vectors X and Y is defined as follows:

$$Cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

  Using the formula for the covariance, the covariance matrix is defined as follows:

$$\Sigma^{n*n} = (c_{i,j}, c_{i,j} = Cov(Dim_i, Dim_j))$$

  Where Dim_x is the xth dimension.
- Then the singular values decomposition (SVD) i.e. the eigenvectors of the covariance matrix are calculated.

$$\Sigma = U * \Lambda * U^{-1}$$

Here Λ is diagonal matrix (with the eigenvalues on the diagonal) and U is an orthogonal matrix of eigenvectors of Σ. These eigenvectors and eigenvalues already provide information about the patterns in the data.
The eigenvector corresponding to the largest eigenvalue is called the first principal component, this component corresponds to the direction with the most variance. Since n eigenvectors can be derived, there are n principal components. They are ordered from high to low based on their eigenvalue. In this way the most relevant principal component are sorted first.

In order to reduce the dimension, we can optionally choose (first) p components, and form a matrix with these vectors in the columns. This matrix is called the feature vector. In the literature, there are several tests known helping in deciding on the number of components to keep.

With these p feature vectors we have two choices. The original data can be transformed to retain only p dimensions, or the noise of the original data set can be reduced using some components while keeping all n dimensions.

2.3 Assumptions and Properties of PCA-Transformed Data

When using this technique we are accepting some ground assumptions behind PCA and correspondingly we have to carefully consider situations where the assumptions are not (fully) valid.

Assumptions of PCA

- Linearity. This fact assumes that new vectors i.e. components are linear combination of original ones, so we rely on the same concept for leakage.
- Components with large variances are the most interesting ones. We show in Sect. 3 that this is not always valid.
- The reduction of the dimension of the original data set does not lead to the loss of important information. On the contrary, this can lead to better results e.g. when noise is removed to improve the key recovery.

As time-domain traces often have multiple samples where leakage is presented, we hypothesize that these samples will be projected onto only a few PCs. This implies that effective filtering strategies are possible when other PCs are filtered out, and additionally CPA could be performed on PCA transformed traces. We investigate this in more details in the remainder of this paper.

2.4 Multiple Leakage Points and PCA

As mentioned above PCA computes new variables i.e. principal components which are derived as linear combinations of the original variables. As a consequence, in the context of power analysis we have the following observation. An interesting property of PCA in the context of trace set analysis is that correlating samples in time are projected onto one or a few PCs. For instance, a PC which has two positive peaks at time t_0 and t_1 implies that the original trace

set has positively correlating values at time t_0 and t_1. Also, the larger the PC is, the larger the variance at these times is.

To show these properties, we analyze applying PCA transformations to some simulated power traces. We first create a noisy trace set with three points of leakage: a point A with leakage, a non-correlating point B with leakage, and a point C with leakage negatively correlating with peak A. Each trace also has some low noise added. Figure 2 shows twenty overlapped traces from this set. We also create a similar set with misaligned peaks (Figure 3). For both trace sets we calculate the principle components, as shown in Figure 4a and 4b.

First, it is clear that the first principle component captures the correlation between the peaks A and C, and the second captures peak B. This also implies that all samples in peak A and C (and similarly peak B) are accumulated into one dimension after the PCA transformation. This can potentially increase DPA leakage, which is calculated per dimension. It is interesting this holds for both aligned and misaligned traces, which also shows that a transformation could project misaligned peaks onto a few dimensions.

Second, in the aligned case, the first two principle components capture all peak information. The other principle components all capture noise. For the misaligned case, the other principle components represent different shifted peaks.

These experiments show that, even under misalignment, PCA transformations can project multiple correlating points of leakage onto several PCs. As DPA on a transformed traceset analyzes PCs separately, this may improve the signal-to-noise ratio.

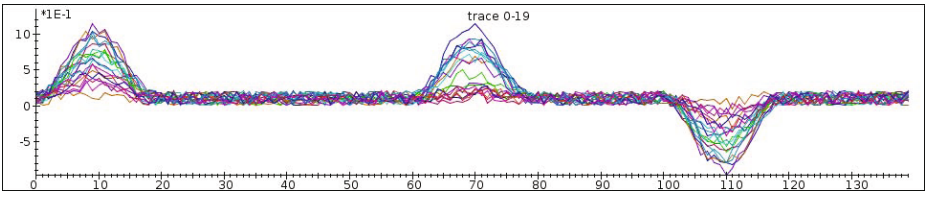


Fig. 2. Aligned traces with three leakage points

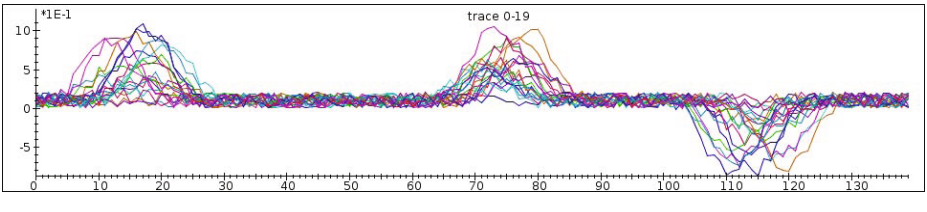
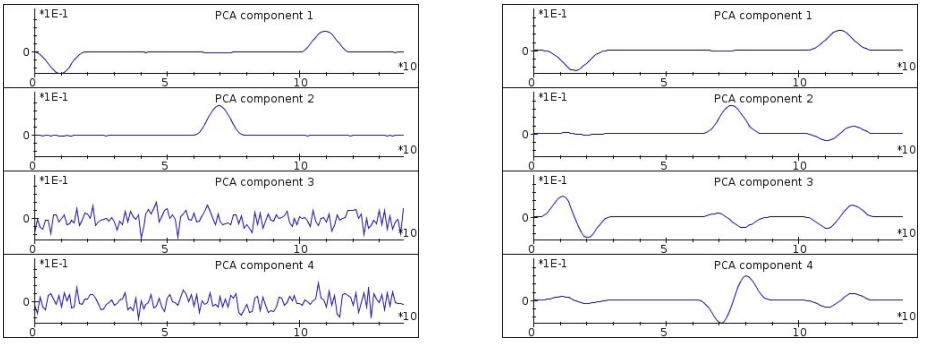


Fig. 3. Misaligned traces with three leakage points



(a) Aligned peaks. (b) Misaligned peaks.

Fig. 4. First four Principle Components of transformed trace sets

2.5 Noise Reduction

Due to various countermeasures that aim at making DPA more difficult by e.g. adding a lot of noise, it is sometimes required to perform a lot of pre-processing to remove the noise for successful key recovery. In particular, one can use chosen Principal Components to retain only certain (sensitive) information. One of the assumptions behind PCA (cf. Sect. 2) is that PC with larger variances represent interesting data, while others with lower variances represent noise. Therefore, the goal is to remove the components which contribute to the noise. The first step is the same as in a transformation, the feature vector U is transposed and multiplied with the transposed mean-adjusted data X.

$$Y = U^T * X^T = (X * U)^T$$

Hence, when extraction the undesired i.e. noise-representing components is performed, the procedure is as follows. The dimensionality of the data is reduced to p by projecting each x in X into $y = U_p^T * x$ where U_p are the first p columns of U, and y is a p-vector. The PCA approximation (of the input x) with only p principal components is then;

$$\tilde{x} = \sum_{j=1}^{p} (u_j^T * x) * u_j$$

and the (squared reconstruction) error can be shown to be equal to $\sum_{i=p+1}^{n} \lambda_i$ that is, the sum of the eigenvalues for the unused eigenvectors.

Choosing the Right Components to Keep. There are extensive discussions in the literature about the choice of components to keep in order to get the maximum from using PCA. The ideas and approaches depend heavily on applications. Since this method is mostly used to find the most distinctive data

(which usually is the data with the most variance), most of the literature deals with deciding about the amount of the "smaller" components that can be left out.

For side-channel analysis, this is not the right route to take. Usually, power traces contain a lot of noise and this noise typically has a large variation relative to the DPA information we are looking to find. This means that, depending on the process of collecting the power measurements to be analyzed, the noise can also be captured by the largest principal components, especially for "real" trace sets i.e. the one where countermeasures are deployed. Since we would like to get rid of the noise, we need to find which principal components we can safely remove without losing the data relating to the secret key. We address this issue in our experiments.

3 Experiments

As described in Section 2, PCA can be used in two ways. We can perform a transformation using PCA and do the analysis on Principal Components, or we could use only a subset of Principal Components to reduce the noise in the original trace set. In this section we address both aspects.

We performed our experiments by taking power measurements of a smartcard which contains a software DES implementation. In order to test PCA against countermeasures, we used an implementation that contains a configurable countermeasure that introduces random delays. We used a Picoscope 5203 and a modified smart card reader to obtain power measurements from the used smartcard. In order to enhance the signal, we used an analog 48 MHz low-pass filter. We performed all experiments also on a hardware DES implementation and on implementations of other cryptographic algorithms i.e. AES and ECC. These experiments showed the same results as described in Section 3, which means that the method is not implementation or algorithm dependent.

3.1 Noise Reduction

We know that the process-related signal within the trace set has a large variation, so it should be captured by the largest Principal Components. Any noise in the measurement is captured by smaller PCs. However, this general observation is not directly applicable to power consumption signals. More precisely, the exact positions of key-related information differ for various implementations and platforms.

Nevertheless, it is valuable to find out where the noise-related information is located, either as a result of some countermeasures or due to measurement set-ups that are used. If we remove these principal components and retain all others (the ones which contain the key information), we might be able to reduce the noise i.e. to improve the signal to noise ratio, and therefore enhance the DPA analysis.

We tested this hypothesis on our set of power measurements of a software DES implementation. We took 200 traces of 312 samples of the DES encryption.

All countermeasures were turned off, which meant that we already could find the correct key using a CPA attack. Also, this meant that we knew in which sample the key leakage was present.

We used this trace set for the rest of our experiments with noise reduction using PCA. We plotted the principal components to see if they contained any interesting properties. It appeared that different principal components captured more or less information for the known key leakage samples. As an example, we inspected the 15th principal component. It appeared to have a high peak for the sample with the key leakage of S-Box 8 so it contained some information about the data at that sample location. We tested this hypothesis by performing a noise reduction retaining all principal components except this 15th. When we performed a DPA attack on the resulting trace set, the correlation for the correct key guess for S-box 8 dropped significantly.

Since we expect the largest amount of non-key information to be captured by the largest principal components, we remove some of these largest components. When we perform a DPA attack on the resulting noise-reduced trace set, we can see an increase in the correlation value for the correct key guess. From this observation, we can conclude that we removed more noise than signal.

We observed that different components capture the DPA information from different S-boxes. This means that the best results are obtained if one knows which component captures the most variation for the sample where the key leakage is. Subsequently this means that for the best results, one needs to know the sample with the key leakage. This implies that one should obtain a card of the same type with a known key to find at which moment in time the information is exploitable. In this way, a sort of profiling is performed i.e. templates are created in order to speed-up the key recovery.

Another useful observation is on software versus hardware implementations. Our findings prove hardware measurements obtained from a card with a co-processor more "noisy" and best results were obtained by removing up to the first 50 components. The exception was a set of measurements obtained from SASEBO-R board where the highest key-dependent leakage was observed within the 3rd principal component. As a conclusion, there is a lot of potential in PCA for noise reduction, but the threshold for improving the leakage has to be decided on the basis of a given implementation. Nevertheless, we were able to improve the leakage in all observed cases.

3.2 PCA Transformation

Whereas during noise reduction we first transform the trace set, remove some components and then transform the trace set back for further analysis, we could also only perform the transformation. This will put the Principal Components on the main axis, which means that all variance that is correlated at different points in time will be projected onto a few PCs as elaborated above.

CPA Highest-Peak Distinguisher. We used the same trace set as before containing 200 traces with 312 samples. To see which effect a transformation has

on the results of a DPA attack, we performed a DPA attack on this transformed trace set. We found that the correct key guess did not contain the highest peak in the correlation graph. This means that we are not able to find the correct key after a PCA transformation.

CPA Abs-Avg Distinguisher. However, when we inspected the correlation graphs for all key guesses, we found that the graph for the correct key guess was different from the graphs for the wrong key guesses, see Fig. 5. The correlation for the first samples (which correspond to the higher principal components) was higher for the correct key guess compared to the wrong key guess. The main difference is however that the correlation for the lower samples was much lower for the correct key guess compared to the wrong key guesses. Actually, the conclusion is that variances are not the same for all PCs, which we expect when the right key is used. This is in line with the results of a normal DPA attack where the correlation for unrelated samples can also be lower for the correct key guess compared to the wrong key guess [10].

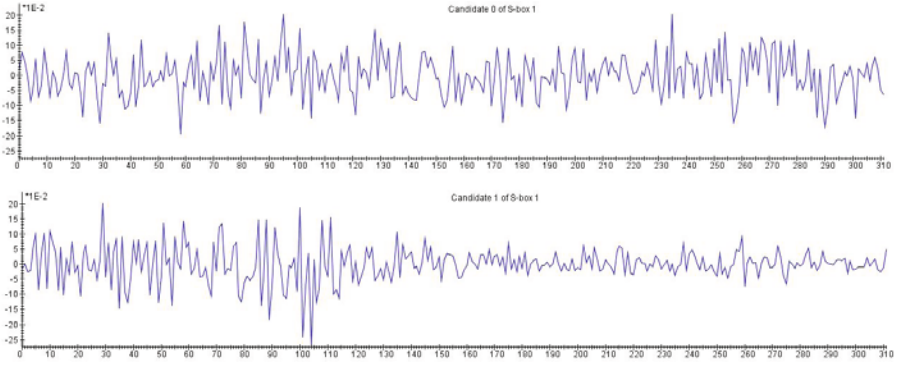


Fig. 5. Correlation trace for the wrong (upper) and the correct (lower) subkey guess

In order to quantify this, we add the absolute values of all samples for all correlation traces x and divide this result by the total number of samples n in order to create an average value avg.

$$avg_x = \frac{\sum_{i=1}^{n} |x_i|}{n}$$

Where x_i denotes the value of the sample at index i in correlation trace x.

We use this method to calculate the absolute average value of each correlation trace for all samples. A plot of these values is shown in Fig. 6. The x-axis shows the key guesses for each S-box i.e. the first 64 values correspond to the 64 subkeys of S-box 1 etc.

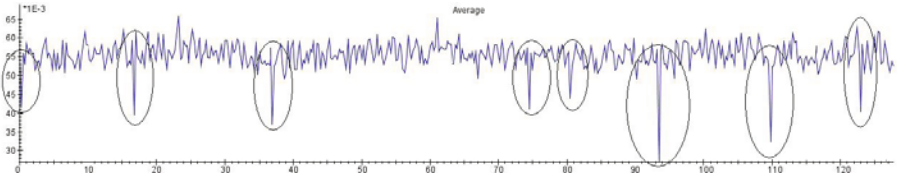


Fig. 6. Absolute average value of each correlation trace for software DES

The main thing we notice in this graph is the 8 outlying peaks at different locations. A peak means that the absolute average value of one correlation trace is lower than the value of the other correlation traces. So we can basically say that the correlation for that key guess is lower than the correlation of the other key guesses. Since DES has 8 subkeys we can easily distinguish these and derive the used secret key. We have found similar results for a hardware DES and an FPGA-implementation of AES-256.

3.3 PCA on Misaligned Traces

An effective countermeasure against DPA attacks is the introduction of random delays during execution of the algorithm. This decreases the effectiveness of DPA as compared to a normal execution since the S-boxes are processed at different moments in time.

Performing DPA on PCA-transformed traces however, is not as sensitive to timing. Misalignment creates, in essence, a correlation between different points in time. This may result in these samples being projected onto a few PCs, and thereby reduce the effect of the random delay countermeasure.

In order to test this hypothesis, we perform a PCA transformation on an obtained trace set of 500 traces with 2081 samples of a smartcard performing software DES with a random delay countermeasure.

We first perform a PCA transformation of the traces. In component 41–57 we find patterns that are interesting; see Fig. 7.

To investigate this further, we perform a DPA attack on the PCA transformed traces and obtain the correlation traces for each key guess. From this we find some peaks at the right key candidate for PC 46, which is very similar to 41. It thereby appears that these components encode and gather the misaligned key information.

When we calculate the absolute average value for each of the obtained correlation traces we get the graph as shown in Fig. 8 (Please note that due to computational issues, we were only able to keep the DPA information for the first five S-boxes.)

From this experiment we find that our hypothesis that misalignment causes a correlation between the neighboring samples that have key leakage, and therefore they are projected onto a few components. After using the absolute-average distinguisher, we are able to fully extract the key of the software DES implementation with random delays.

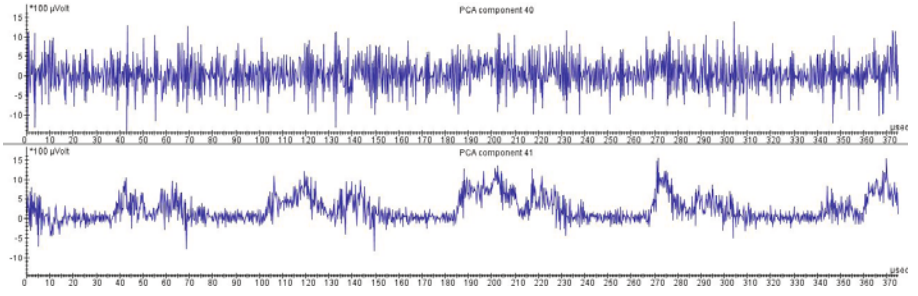


Fig. 7. The 40th and the 41 principal component of a PCA transformed traceset with random delays

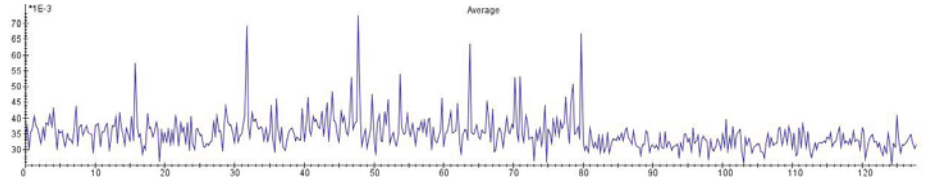


Fig. 8. Absolute average value of each correlation trace for software DES

4 Comparison to Other Alignment Techniques

There are several other algorithms proposed to handle the misalignment countermeasure e.g. [9,16,11] and in this Section we compare PCA with one of them i.e. with Static alignment.

Static alignment is the most natural method for the treatment of misaligned traces and it is clearly described in [9] by Mangard et al. To apply the algorithm, it is first required to choose a fragment in a so-called reference trace, which should be ideally close to the attacking interval. Then the algorithm searches for the same fragment in the other traces and shifts them accordingly. In this way the alignment of the reference fragment is performed. The main disadvantages of this method is in somewhat reduced efficiency, when compared to more recent algorithms but it does improve on the number of traces required for a successful DPA attack.

We compare static alignment with PCA on the height of the peak for the correct key guess. For both methods, we compare the difference in the height of the peak i.e. the correlation values for the correct key guess and for the first wrong key guess. For PCA, we actually look at the difference in the height of the average value of the correlation trace. To derive the values for static alignment, we use the misaligned trace set from our sample card and statically align them before doing a DPA attack. For PCA, we use the same (misaligned) trace set, to which we first perform a PCA transformation, and afterwards we calculate the

Table 1. Comparison between Static alignment and PCA

	Static alignment	PCA
Correct key guess	0.4035	0.0450
First wrong key guess	0.2869	0.0393
Difference	28.9%	12.7%

absolute average value for 150 samples of the correlation traces. The results can be found in Table 1.

We see that PCA does not outperform static alignment, at least for the chosen trace set. However, the results should be considered less strictly as the method used for PCA differs from the one for static alignment i.e. actual correlation values versus absolute average values. Nevertheless, it proves PCA a viable method for alignment along with pre-processing. As future work, we plan to perform a meaningful comparison with other, recently published alignment methods such as Elastic alignment [16] and RAM [11].

5 Conclusions

In this work we introduce Principal Component Analysis as a suitable preprocessing technique on the power traces to enhance the effectiveness of DPA attacks. In particular, we advocate two separate cases to use PCA, for noise reduction and a PCA transformation (before the actual DPA). Our results are verified in practice by several experiments on both, protected and unprotected implementations. In the experiments we were able to improve the signal to noise ratio in various occasions when the location of the key leakage is known. We were able to de-noise a given trace set by retaining only the Principal Components which capture the variance at the location of the key leakage. The effect of this noise reduction was that the guessed, correct subkeys had a higher correlation when a DPA attack was performed on the noise-reduced trace set as opposed to the correlation on the original trace set. This method works for each of the trace sets we used.

Acknowledgements. We would like to thank Riscure for providing an environment for fruitful discussion during the research, and for providing the side-channel analysis platform that was used for this work (Inspector). We also thank Yang Li and Kazuo Sakiyama from University of Electro Communication, Tokyo for providing us with suitable traces from a SASEBO board. We thank Elena Marchiori from RU Nijmegen for her insightful comments.

This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State and by the European Commission under contract number ICT-2007-216676 ECRYPT NoE phase II and by the K.U.Leuven-BOF (OT/06/40).

References

1. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template Attacks in Principal Subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
2. Bohy, L., Neve, M., Samyde, D., Quisquater, J.-J.: Principal and independent component analysis for crypto-systems with hardware unmasked units. In: Proceedings of e-Smart 2003 (2003)
3. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
4. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
5. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis - A Generic Side-Channel Distinguisher. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
6. Hotelling, H.: Analysis of a complex of statistical variables into principal components. The Journal of Educational Psychology, 417–441 (1933)
7. Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer Series in Statistics. Springer, New York (2002)
8. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
9. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag New York, Inc., Secaucus (2007)
10. Messerges, T.S.: Power analysis attacks and countermeasures for cryptographic algorithms. PhD thesis, University of Illinois at Chicago, Chicago, IL, USA (2000)
11. Muijrers, R.A., van Woudenberg, J.G.J., Batina, L.: RAM: Rapid Alignment Method. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 266–282. Springer, Heidelberg (2011)
12. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine Series 2(6), 559–572 (1901)
13. Smith, L.I.: A tutorial on principal components analysis (February 2002), http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
14. Souissi, Y., Nassar, M., Guilley, S., Danger, J.-L., Flament, F.: First Principal Components Analysis: A New Side Channel Distinguisher. In: Rhee, K.-H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 407–419. Springer, Heidelberg (2011)
15. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
16. van Woudenberg, J.G.J., Witteman, M.F., Bakker, B.: Improving Differential Power Analysis by Elastic Alignment. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 104–119. Springer, Heidelberg (2011)

An Efficient Protocol for Oblivious DFA Evaluation and Applications

Payman Mohassel[1], Salman Niksefat[2,⋆],
Saeed Sadeghian[3], and Babak Sadeghiyan[4]

[1] University of Calgary
pmohasse@cpsc.ucalgary.ca
[2] Amirkabir University of Technology
niksefat@aut.ac.ir
[3] University of Calgary
sadeghis@ucalgary.ca
[4] Amirkabir University of Technology
basadegh@aut.ac.ir

Abstract. In this paper, we design an efficient protocol for *oblivious DFA evaluation* between an input holder (client) and a DFA holder (server). The protocol runs in a single round, and only requires a small amount of computation by each party. The most efficient version of our protocol only requires $O(k)$ asymmetric operations by either party, where k is the security parameter. Moreover, the client's total computation is only linear in his own input and independent of the size of the DFA. We prove the protocol fully-secure against a *malicious client* and *private* against a malicious server, using the standard *simulation-based* security definitions for secure two-party computation.

We show how to transform our construction in order to solve multiple variants of the *secure pattern matching* problem without any computational overhead. The more challenging variant is when parties want to compute the number of occurrences of a pattern in a text (but nothing else). We observe that, for this variant, we need a protocol for counting the number of accepting states visited during the evaluation of a DFA on an input. We then introduce a novel modification to our original protocol in order to solve the counting variant, without any loss in efficiency or security.

Finally, we fully implement our protocol and run a series of experiments on a client/server network environment. Our experimental results demonstrate the efficiency of our proposed protocol and, confirm the particularly low computation overhead of the client.

1 Introduction

In the oblivious Deterministic Finite Automata (DFA) evaluation problem, the first party (Server) holds a DFA Γ, while the second party (Client) holds an input string X. Their goal is to collaboratively evaluate the DFA Γ on input

⋆ Work done while visiting University of Calgary.

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 398–415, 2012.

X, allowing one or both of the participants to learn the result $\Gamma(X)$ without learning any additional information about each other's input. A number of applications with security and privacy concerns can be efficiently formulated as DFA evaluation and be implemented using secure two-party protocols for oblivious DFA evaluation.

One such example is the problem of *secure pattern matching* (or text processing) and its variants which have been the focus of several recent works in the literature [15,2,9,3,6]. In the most common variant of the problem, one is interested in finding the locations of a specific pattern p in a text T. Pattern matching has immediate applications in mining and processing DNA data and is often used in practice, e.g. in the Combined DNA Index System (CODIS)[1] run by the FBI for DNA identity testing. There are privacy concerns associated with algorithms that process individual's DNA data and, not surprisingly, privacy issues are the main motivation behind most of the above-mentioned works on designing secure solutions.

One can formulate the basic variant of the pattern matching problem as the evaluation of a *pattern-specific* automaton Γ_p on a text T [10]. In fact, several of the papers mentioned above solve the secure pattern matching problem by designing protocols for oblivious evaluation of Γ_p on T [3,15,2].

Depending on the application being considered, it can be the case that the size of the input string to the DFA is large (e.g. the text T in secure pattern matching), or the size of the DFA Γ itself (e.g. when many patterns are combined into one DFA). Therefore, *for an oblivious DFA evaluation protocol to be a viable solution for practice, it needs to ensure efficiency and scalability when run on large DFAs and/or input strings.* Towards this goal, we focus on the following three efficiency criteria:

- **Small Number of Asymmetric Operations:** Based on existing benchmarks (e.g. `http://bench.cr.yp.to`) asymmetric operations (e.g. exponentiation) require *several thousand* times more cpu cycles compared to their symmetric-key counterparts. Hence, for an ODFA protocol to be scalable for large input strings and large DFA sizes, it is essential to minimize the number of asymmetric operations and to ensure that their number does not grow with the size of the DFA and/or its input. ODFA protocols of [15] and [3] do not satisfy this property since the number of exponentiations they require is linear in the size of the DFA and its input.
- **Small Computation for the Input Holder (client):** In practice, the two involved parties do not always have the same computational resources, and hence it is common to implement the protocols in a client/server model where one party has to perform noticeably less work. Motivated by this concern, we require that the input holder's (client) total work be significantly smaller, and in particular be independent of the size of the server's DFA. All previous solutions for ODFA, including a general solution based on Yao's garbled circuit protocol fail to achieve this goal.

[1] `http://www.fbi.gov/hq/lab/codis`

– **Small Number of Rounds of Interaction:** we also require our protocols to have a small number of rounds of interactions (ideally a single round). A *single round* of interaction allows the protocol to be deployed in a non-interactive setting where one party can communicate his message, go offline and connect at a later time to retrieve the final message. Therefore, very little online coordination and computation is necessary.

As mentioned above, the existing solutions for oblivious DFA evaluation do not meet one or more of the above efficiency criteria.

1.1 Our Contributions

A New Protocol For Oblivious DFA Evaluation. Our main contribution is a new and efficient protocol for oblivious DFA evaluation that meets all three of the above-mentioned efficiency criteria. The most efficient variant of our construction runs in one and a half rounds, and only requires $O(k)$ asymmetric operations by either party where k is the security parameter. Moreover, the input holder's total work is only linear in his input and is independent of the DFA size.

We prove the protocol fully-secure against a *malicious client* and *private* against a *malicious server*, using the standard *simulation-based* security definitions for secure two-party computation.

Our starting point is a single round protocol between a server who holds the DFA Γ with $|Q|$ states and a client who holds an n-bit input string X. The basic idea is for the server to represent the evaluation of Γ on an arbitrary n-bit string X via a $n \times |Q|$ DFA matrix M_Γ. A DFA matrix is a simple data structure used to efficiently evaluate the DFA on any input string of size n. The server *permutes* and *garbles* this matrix into a garbled DFA matrix GM_Γ and sends it to the client.

After the garbling stage, the server and the client engage in a series of oblivious transfer protocols where the client learns a vector of random keys corresponding to his input X. These random keys allow the client to ungarble a unique path that starts from the first row of the matrix and ends in the last row. This path (referred to as the *transit path*) corresponds to the evaluation of input X using the DFA matrix M_Γ. The client can extract the final output of evaluation from this ungarbled transit path but is not able to ungarble any of the remaining elements in the matrix, or learn any additional information about the DFA.

The number of OTs can be made independent of the client's input size (i.e., the number of OTs remains the same, as the input size increases) via use of the OT extension technique of [7]. More precisely, this extension reduces the number of exponentiations necessary from $O(n)$ to $O(k)$, but increases the number of rounds from a single round to one and a half round.

Comparison with Yao's protocol. We note that the above approach for DFA evaluation is reminiscent of the Yao's garbled circuit protocol [16,11], where the circuit being evaluated is garbled and a set of random keys are used to ungarble and evaluate the circuit on a specific input. In fact, it is possible to

use Yao's garbled circuit protocol to implement oblivious DFA evaluation. One party's input to the circuit is his input string while the other party's input is the DFA itself. However, as discussed in [3], the resulting protocol would be significantly less efficient compared to ours. Moreover, unlike our construction, an implementation of ODFAs via a direct application of Yao's garbled circuit would yield a protocol wherein the amount of work the client has to perform is linear in the size of the circuit and hence at least linear in the size of the DFA. Such a protocol would not satisfy our second efficiency criteria.

However an alternative way of presenting our construction is to describe it as a generalization of Yao's garbled circuit protocol where the gates are allowed to take non-boolean inputs and return non-boolean outputs. We discuss this variant in more detail, in Section 4.5.[2]

We give a more detailed comparison of efficiency between our protocol and the existing solutions in Section 4.6.

Applications to Secure Pattern Matching. We show how to use our Oblivious DFA evaluation protocol to efficiently solve multiple variants of the *Secure Pattern Matching* problem. In the three main variants we consider, one party holds a text T while the other party holds a pattern p and the aim is for the first party to learn one of the following but nothing else: (i) whether or not p appears in T, (ii) all the locations (if any) where p occurs as a pattern in T, or (iii) the number of occurrences of pattern p in T, while the text holder learns nothing about the pattern.

The first two variants can be implemented in a relatively straightforward manner, using appropriate pattern-specific DFAs. In the third variant, we need to count the number of occurrences of a pattern p in a text T. As discussed in [9], the number of occurrences of a pattern p is in fact what some applications of pattern matching are interested in. It is not clear how to directly cast this problem as a DFA evaluation problem and unlike the existing solutions for the second variant, we need to hide the locations where the patterns occur from both parties. It is not obvious how to modify any of the existing secure pattern matching constructions to solve this variant of the problem without a noticeable increase in complexity.

To design an efficient protocol for this task, we show how to modify our oblivious DFA evaluation protocol so that it returns the total number of times that accepting state(s) are visited during the evaluation of an input, instead of a single bit indicating an accept/reject final state. In particular, we embed a series of "random looking but correlated" values in the DFA matrix before garbling it and show how to modify the original protocol to let the evaluator of the garbled DFA matrix recover all the *embedded strings* on the transit path. The evaluator then uses these values to compute the number of accepting states visited without learning any additional information. The resulting protocol's complexity is similar to our original ODFA construction. When applied to the pattern-specific DFA of [10], our construction automatically yields a secure protocol for counting

[2] This was pointed out by one of the reviewers of our paper.

the number of occurrences of a pattern p in a text T. This new variant of ODFA maybe be of independent interest in other applications as well.

Implementation and Experimental Results. We fully implement our main ODFA protocol in a client/server network environment and use the OT extension of [7] to implement the oblivious transfer component. We measure the performance of our implementation for a wide range of input and DFA sizes. Experiments are ran on two machines as the client and the server, each with an Intel Core i7 processor with 4GB of RAM and connected via a Gigabit Ethernet. Our experiments confirm our theoretical arguments on the scalability of our protocol. For instance, on 20-bit inputs and for DFA sizes of as large as 15000 states, or for DFAs with 20 states and inputs as large as 10000-bits, our protocol runs in less than 1 second. These numbers remain fairly low (under 12 seconds) even when we increase the number of states or the input bits to 150000. Our experiments show that the client's computation is very low, such that for the case of a DFA with 20 states and inputs of size 150000 bits, his computation hardly reaches 1 second. For the case of 20-bit inputs and 150000 state DFAs, client's computation is even smaller (less than 32 milliseconds). This confirms the suitability of our protocol for client/server settings, where the input holder has limited computational resources.

We also note that since we use the OT extensions of [7], OTs are no longer the computational bottleneck for the server. The main bottleneck for large inputs and DFAs is the computation the server performs to garble the DFA matrix. A more detailed discussion of the implementation and the results of experiments are given in Section 6.

1.2 Related Work

To the best of our knowledge, the first scheme for *oblivious DFA evaluation* was proposed in [15] (motivated by the problem of privacy preserving DNA pattern matching). Their construction is not constant round and only provides security against semi-honest adversaries. This work was later improved by Frikken [2] who designed a protocol, with security against semi-honest adversaries, that runs in a constant number of rounds (more than one) and has fewer asymmetric computation (exponentiation).

[3] is the only work on oblivious DFA evaluation that considers malicious adversaries but requires $\min(O(|Q|), O(n))$ rounds of interaction and $O(n|Q|)$ asymmetric computations, where n is the input size and $|Q|$ is the number of states in the DFA. The security of our protocol against the input holder is similar to that of [3], but we achieve a weaker notion of security against a malicious DFA holder (see Section 4 for more detail). It is also possible to use Yao's garbled circuit protocol to implement oblivious DFA evaluation, but as discussed above, the resulting protocol would not satisfy our efficiency criteria.

The problem of oblivious DFA evaluation can also be formulated as computation on encrypted data and be implemented using the construction of [8] for

branching programs or the recent fully homomorphic encryption schemes [4]. The problem with these schemes is their high computation cost as the number of times the corresponding public-key encryption schemes are invoked is at least linear in the DFA size and its input.

See Table 1 for a more detailed comparison of our protocol with the existing solution for oblivious DFA evaluation.

We also briefly review the status of protocols for *secure pattern matching* here. Let n be the text size and m be the pattern size. The protocol of [3] runs in $O(m)$ rounds and requires $O(mn)$ exponentiations. The constructions of [5] and [6] run in a constant number of rounds (more than one) and require $O(n+m)$ exponentiations. For long texts, where n is large, this renders the exponentiations a major computational overhead. In contrast, an extended version of our protocol (in random oracle model) only requires $O(k)$ exponentiations where k is the security parameter. This improves the efficiency significantly when $n \gg k$.

2 Preliminaries

In this section, we introduce the notations used in the rest of the paper. The cryptographic primitives as well as the security definitions are omitted due to lack of space. Readers are referred to the full version [12] for more detail.

2.1 Notations

Throughout the paper, we use k to denote the security parameter. We denote an element at row i and column j of a matrix by $M[i, j]$. If the element itself is a pair we use $M[i, j, 0]$ to denote the first value of the pair and $M[i, j, 1]$ to denote the second value. Vectors are denoted by bold-face letters such as $\boldsymbol{v}$. We use $a||b$ to denote the concatenation of the strings a and b. λ is used to denote an empty string and a^b denotes b consecutive concatenation of the string a by itself.

We denote a random permutation function by $Perm$. $\boldsymbol{v} \leftarrow Perm(Q)$ takes as input a set of integers $Q = \{1, \ldots, |Q|\}$, permutes the set uniformly at random and returns the permuted elements in a row vector v of dimension $|Q|$. We call a matrix a *permutation matrix* if all of its rows are generated in this way. The following simple algorithm (algorithm 1) can be used to generate a permutation matrix PER with n rows from the set Q.

Algorithm 1. GenPerm(n, Q)

for $1 \leq i \leq n$ **do**
  $PER[i] \leftarrow Perm(Q)$
end for
return PER

3 DFA and Its Matrix Representation

3.1 DFA

In this paper a deterministic finite automaton (DFA) [14] is denoted by a 5-tuple $\Gamma = (Q, \Sigma, \Delta, s_1, F)$, where Q is a finite set of states, Σ is a finite input alphabet, $s_1 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Δ denotes the transition function and $|Q|$ denotes the total number of states. We represent states by integers in $\mathbb{Z}_{|Q|}$. $\Delta(j, \alpha)$ returns the next state when the DFA is in state $j \in Q$ and sees an input $\alpha \in \Sigma$. A string $X = x_1 x_2 \dots x_n \in \Sigma^n$ is said to be accepted by Γ if the state $s_n = \Delta(\dots \Delta(\Delta(s_1, x_1), x_2) \dots, x_n)$ is a final state $s_n \in F$. A binary DFA is a DFA with $\Sigma = \{0, 1\}$. From this point forward, we restrict our attention to binary DFAs and the term DFA is used for binary DFAs.

Our oblivious evaluation protocols take advantage of a *matrix representation* of DFAs. Next we define the notions of a *DFA matrix* and a *permuted DFA matrix* which we use throughout the paper.

3.2 DFA Matrix

Assume that the input string of a DFA $\Gamma = (Q, \{0, 1\}, \Delta, s_1, F)$ is a bitstring $X = x_1 x_2 \dots x_n \in \{0, 1\}^n$. Then we can represent the evaluation of Γ on an arbitrary input X of length n as a matrix M_Γ of size $n \times |Q|$. For $1 \leq i \leq n$, the ith row of M_Γ represents the evaluation of x_i.

In particular, the element $M_\Gamma[i, j]$ stores the pair $(\Delta(j, 0), \Delta(j, 1))$ which encodes the indices of the next two states to be visited (at row $i + 1$) for input bits $x_i = 0$ and $x_i = 1$, respectively. At row n where the last bit x_n is processed, instead of storing the indices of the next states, we put a 1 if the next state is an accepting one, and a 0 otherwise.

There is a simple algorithm for converting a DFA to its corresponding DFA matrix representation. In the rest of the paper, we denote this algorithm with $\mathsf{DfaMat}(\Gamma, n)$, which takes a DFA Γ, and an input string of size n as its input and generates the DFA matrix M_Γ.

Evaluation Using the DFA Matrix. One can use M_Γ to efficiently evaluate Γ on any n bit input X. We start at $M_\Gamma[1, 1]$. If $x_1 = 0$, the first index of the pair $M_\Gamma[1, 1]$ is used to locate the next cell to visit at row 2. If $x_1 = 1$, the second index of $M_\Gamma[1, 1]$ is used instead. Then, by considering the chosen pair in row 2 and the value of x_2, one can find the next pair to visit in row 3. This process is repeated until we reach row n and read either 0 or 1 which will be the result of the evaluation of X on Γ.

When evaluating an input string X using a DFA matrix, we call the set of pairs visited starting from row 1 upto row n a *transit path* for X. A transit path either ends with 1 which shows that X is accepted by Γ or ends with 0 which shows that X is not accepted by Γ.

3.3 Permuted DFA Matrix

A permuted DFA matrix PM_Γ is generated by randomly permuting the elements in each row i of M_Γ and updating the associated indices in row $i-1$ accordingly to point to the new permuted indices of row i. In order to do this, we first generate a permutation matrix PER of size $n \times |Q|$ using the GenPerm algorithm 1. There is a simple algorithm that takes PER and M_Γ and converts a DFA matrix M_Γ to an equivalent permuted DFA Matrix PM_Γ (See [12] for details). In the rest of the paper, we refer to this algorithm as PermDfaMat().

Evaluating an input using the permuted DFA matrix is almost identical to the normal DFA matrix with the exception that the evaluation begins at $PM_\Gamma[1, PER[1, 1]]$.

4 An Efficient Protocol for Oblivious DFA Evaluation

Let the server hold a *private* deterministic finite automata (DFA) $\Gamma = (Q, \{0,1\}, \Delta, s_1, F)$ and the client hold a private string $X = x_1 x_2 \dots x_n \in \{0,1\}^n$. Our goal is to let the client discover whether his string X is accepted by the server's DFA Γ or not without revealing anything about X and Γ to server and client, respectively. In this section we propose a new and efficient protocol for oblivious DFA evaluation.

The main version of our protocol is a single-round construction that only requires $O(n)$ exponentiations for both the server and the client, which is independent of the size of DFA. In the random oracle model and using the OT extension of [7], we can make this number independent of client's input by further reducing the number of exponentiations to $O(k)$, where k is the security parameter (at the cost of adding an extra round). In situations where $n \gg k$ which is the case in many applications of DFAs in practice, this leads to a noticeable improvement in efficiency.

We prove the security of our proposed protocol using the standard simulation-based definitions of security for two-party computation.

4.1 A High Level Overview

Before describing our protocol in more detail we start with a high level overview.

Client Gets His Input Keys. For every bit of client's input x_i, server and client engage in an oblivious transfer where server's inputs are two random key strings (K_i^0, K_i^1) corresponding to input bit values 0 and 1. As a result client learns one of the keys in each pair.

Server Computes a Garbled DFA Matrix. In this stage, server (the holder of the DFA Γ) first computes a permuted DFA matrix PM_Γ corresponding to her DFA by calling DfaMat(), GenPerm() and PermDfaMat() algorithms (See Section 3). The permutations are done for the purpose of hiding the structure of the DFA from client.

Server then garbles the permuted DFA matrix in a special way. To garble the matrix server first generates a $n \times |Q|$ matrix PAD filled with random strings. Consider a pair (a_0, a_1) stored in the cell $PM_\Gamma[i, j]$ of the permuted matrix. Each value in the pair is encrypted using a one-time pad encryption where the pad is a combination of $PAD[i, j]$ and the input key strings K_i^0, and K_i^1. More specifically, a_0 is encrypted using K_i^0 while a_1 is encrypted using K_i^1. Then, the resulting ciphertexts are concatenated and encrypted using $PAD[i, j]$ as the seed to the PRG G. All the encryptions are one-time pad encryptions.

Note that client can only decrypt a_b if he knows both the correct input key K_i^b and the random string $PAD[i, j]$. Client will learn one of the two input keys through the oblivious transfer, but this is not sufficient for decrypting either value in the pair. Client learns $PAD[i, j]$ only if he is visiting from a legitimate previous state in the DFA. In order to enforce the latter, $PAD[i, j]$ is concatenated to the appropriate value (i.e. index) already stored in $PM_\Gamma[i-1, j']$, where j' is the permuted index (at row $i-1$) of a legitimate previous state. It is only after this concatenations that the matrix is garbled using the one-time pads described above.

Server sends the resulting garbled DFA matrix GM_Γ plus the index and the pad of starting cell in row 1 to the client. Note that the PAD matrix is not sent to the client.

Client Evaluates the Garbled DFA Matrix. Client uses the input keys he retrieves at the OT stage, to decrypt one of the two values in the starting pair. As a result, he learns the index to a single pair in the next row in addition to a random pad that he uses to partially decrypt the values in that pair. He then decrypts exactly one of the values in the pair (completely) using the retrieved key for his second input bit.

He repeats this process, moving along the *transit path* for input X until he reaches the last row and recovers the final output. First, note that for all the elements not on his transit path, client does not learn the corresponding random string in the PAD matrix and hence those elements remain garbled to him. For those pairs that appear on his path, he can only decrypt one of the two values using the single input key he has retrieved at the OT stage. This rough intuition behind the security against a malicious client is formalized in the security proof.

4.2 The Protocol 1

Server's Input: A DFA $\Gamma = (Q, \{0,1\}, \Delta, s_1, F)$.
Client's Input: A bitstring $X = x_1 x_2 ... x_n \in \{0,1\}^n$.
Common Input: The security parameter k, the OT security parameter κ and the size of DFA $|Q|$. We let $k' = k + \log|Q|$ throughout the protocol. Parties also agree on a 1-out-of-2 OT protocol $OT = (\mathrm{G_{OT}}, \mathrm{Q_{OT}}, \mathrm{A_{OT}}, \mathrm{D_{OT}})$ and a PRG $G : \{0,1\}^k \to \{0,1\}^{2k'}$.

1. **Client encrypts his inputs using OT queries, and sends them to server.**

Sending OT Queries to server

Client computes $(pk, sk) \leftarrow \mathrm{G}_{\mathrm{OT}}(1^{\kappa})$
for $1 \leq i \leq n$ **do**
  client computes $q_i \leftarrow \mathrm{Q}_{\mathrm{OT}}(pk, 1^2, 1^{k'}, x_i)$
end for
Client sends pk and $\boldsymbol{q} = (q_1, q_2, \ldots, q_n)$ to server.

2. **Server Computes a Garbled DFA matrix GM_{Γ}.**

Generating random pads and a permuted DFA matrix PM_{Γ}

Server generates n random key pairs for the OTs:
for $1 \leq i \leq n$ **do**
  $(K_i^0, K_i^1) \xleftarrow{\$} \{0,1\}^{k'}$
end for
Server generates a random pad matrix $PAD_{n \times |Q|}$:
for $i = 1$ to n and $j \in Q$ **do**
  $PAD[i,j] \xleftarrow{\$} \{0,1\}^{k}$
end for
server generates a DFA matrix M_{Γ}:
$M_{\Gamma} \leftarrow \mathsf{DfaMat}(\Gamma, n)$
server generates a random permutation matrix $PER_{n \times |Q|}$:
$PER \leftarrow \mathsf{GenPerm}(n, Q)$
server generates a permuted DFA permuted matrix PM_{Γ}:
$PM_{\Gamma} \leftarrow \mathsf{PermDfaMat}(M_{\Gamma}, PER)$

Computing the Garbled DFA Matrix GM_{Γ} from PM_{Γ}

for each row $i = 1$ to n **do**
  for each $j \in Q$ **do**
    if $1 \leq i \leq n-1$ **then**
      $GM_{\Gamma}[i,j,0] \leftarrow PM_{\Gamma}[i,j,0] || PAD[i+1, PM_{\Gamma}[i,j,0]]$
      $GM_{\Gamma}[i,j,1] \leftarrow PM_{\Gamma}[i,j,1] || PAD[i+1, PM_{\Gamma}[i,j,1]]$
    else if $i = n$ **then**
      $GM_{\Gamma}[n,j,0] \leftarrow (PM_{\Gamma}[n,j,0])^{k'}$
      $GM_{\Gamma}[n,j,1] \leftarrow (PM_{\Gamma}[n,j,1])^{k'}$
    end if
    $GM_{\Gamma}[i,j,0] \leftarrow GM_{\Gamma}[i,j,0] \oplus K_i^0$
    $GM_{\Gamma}[i,j,1] \leftarrow GM_{\Gamma}[i,j,1] \oplus K_i^1$
    $pad_0 || pad_1 \leftarrow G(PAD[i,j])$
    $GM_{\Gamma}[i,j,0] \leftarrow GM_{\Gamma}[i,j,0] \oplus pad_0$
    $GM_{\Gamma}[i,j,1] \leftarrow GM_{\Gamma}[i,j,1] \oplus pad_1$
  end for
end for

3. **Server computes the OT answers $\boldsymbol{a}$, and sends $(\boldsymbol{a}, GM_\Gamma, PER[1,1], PAD[1, PER[1,1]])$ to client.**

Sending OT Answers and the Garbled Matrix to client

for $1 \leq i \leq n$ **do**
  $a_i \leftarrow \mathrm{A}_{\mathrm{OT}}(pk, q_i, K_i^0, K_i^1)$
end for
Server sends $(\boldsymbol{a}, GM_\Gamma, PER[1,1], PAD[1, PER[1,1]])$ to client where $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$.

4. **Client retrieves the keys and computes the final result.**

Computing the Final Output

$state \leftarrow PER[1,1]$
$pad \leftarrow PAD[1, PER[1,1]]$
for $i = 1$ to $n-1$ **do**
  $K_i^{x_i} \leftarrow \mathrm{D}_{\mathrm{OT}}(sk, a_i)$
  $pad_0 || pad_1 \leftarrow G(pad)$
  $newstate || newpad \leftarrow K_i^{x_i} \oplus pad_{x_i} \oplus GM_\Gamma[i, state, x_i]$
  $pad \leftarrow newpad$
  $state \leftarrow newstate$
end for
$pad_0 || pad_1 \leftarrow G(pad)$
Client outputs $GM_\Gamma[n, state, x_n] \oplus pad_{x_n} \oplus K_n^{x_n}$ as his final output.

It is easy to verify that if both parties behave honestly, the protocol correctly evaluates server's DFA Γ on client's input X. In particular, client has the secret information necessary to decrypt one of the two values in each pair on the transition path for input X (in the garbled DFA matrix). Next, we focus on the proof of security of the protocol and a careful analysis of its efficiency.

4.3 Security Proof

We show that as long the oblivious transfer protocol used is secure, so is our protocol. Particularly, if the OT is secure against malicious (semi-honest) adversaries when executed in parallel, our oblivious DFA evaluation protocol described above is also secure against malicious (semi-honest) adversaries. The following Theorem formalizes this statement. See the full version of the paper [12] for the proof.

Theorem 1. *In the OT-hybrid model, and given a computationally secure PRG G, the above protocol is* fully-secure *against a malicious client (see definition 1 of [12]) and is* private *against a malicious server (see definition 2 of [12]).*

4.4 Using OT Extension

In our protocol, the main computational overhead for the client is the $O(n)$ exponentiations required for invoking $n \times OT_1^2$. However, using the extended

OT protocol of [7] we can reduce the number of exponentiations from $O(n)$ to $O(k)$ at the expense of security in the *random oracle model*. This improvement is significant in those applications of oblivious DFA evaluation where $n \gg k$. This is particularly the case in the secure pattern matching applications where n represents the size of the text being searched which is often rather large. Using the OT extension also leads to a slight increase in the number of transferred messages (from 2 to 3). In other words, the number of rounds increase from 1 to 1.5.

4.5 A Different Presentation of Our Protocol

An alternative presentation of our construction is to describe it as a generalization of Yao's garbled circuit protocol, where the gates to the circuit can take non-boolean inputs, and return non-boolean outputs.[3]

More specifically, one can evaluate a DFA D with Q states on a (boolean) input string $x = x_1...x_n$ by repeatedly evaluating a "gate" g that takes as input the current state q_i after reading the first i bits of x (so q_0 is just the initial state) and x_i and outputs the next state q_{i+1}. After n applications of the gate g, we obtain the final state q_n of the DFA (explicitly, $q_n = g(g(...g(q_0, x_1), ...), x_n)$), and then we add one more gate to check whether q_n is an accepting state or not. One can generalize Yao's garbled circuit construction to handle such non-boolean gates. In particular, each gate is garbled by constructing $2Q$ ciphertexts, two for each row. Similar to Yao's protocol, each ciphertext is a "double-key" encryption where one of the keys determines x_i's value and the other determines the input state q_i (in each gate g, a unique key is assigned to each state). Each ciphertext encrypts the key for the next state which is determined using the transition function. Hence, each garbled gate g contains $O(|Q|)$ ciphertexts, and requires $O(|Q|)$ symmetric-key operations by the server to compute. Note that the ciphertexts also need to embed the (after permutation) index of the next row of ciphertexts to consider in the upcoming gate. With this approach, the circuit evaluator only needs to perform $O(1)$ symmetric-key operations per gate to decrypt the output key for each gate.

4.6 Efficiency

In this section we present the complexity analysis of our basic protocol.

Rounds of Communication: Our protocol runs in one round which consists of a message from client to server and vice versa.

Asymmetric Computation: We have tried to minimize the number of required asymmetric computation in our protocol since asymmetric operations are significantly more expensive. The only asymmetric computation we perform in our protocol is for the OTs. Since each OT requires a constant number of exponentiations and there are n invocations of such OTs, the overall number of exponentiation in our protocol is bounded by $O(n)$ for both the server and the

[3] This presentation was pointed out to us by one of the reviewers of our paper at CT-RSA 2012.

client. Using the amortized OT protocol of Naor and Pinkas [13], server and client have to perform one and two exponentiations per OT, respectively. Our use of OT extension further reduces this bound to $O(k)$, where k is the security parameter.

Symmetric Computation: The only symmetric computation in our protocol is the PRG invocations. Server performs $2n|Q|$ PRG invocations to build GM_Γ, so the symmetric computation for the server is $O(n|Q|)$. Client performs n PRG invocations for computing the final output and hence the number of symmetric operations by the client is only $O(n)$.

Communication Complexity: The communication complexity of the protocol is dominated by the number of bits stored in the garbled DFA matrix GM_Γ which is bounded by $O(n|Q|k)$ where k is the security parameter.

Comparison to Previous Work: Table 1 summarizes and compares the computational and communication costs of our proposed protocol with the related work. The complexity for a Yao's-based construction is borrowed from the analysis given in [3]. The complexity of the ODFA protocol based on the construction of [8] is derived by considering a branching program corresponding to evaluation of a input of size n on a DFA of size Q. Note that in all the existing constructions except for the one base on Yao's garbled circuit protocol, the number of asymmetric operations (exponentiations) by the server is at least linear in both the input size n and the DFA size Q. In our protocol, on the other hand, this number is $O(n)$ in the standard model and $O(k)$ in the random oracle model. This is a significant improvement in efficiency when dealing with large DFA sizes. Another efficiency criteria we are interested in is small computation by the input holder (client). In all the previous constructions except for the one based on [8], the client's work is at least linear in the DFA size which is undesirable in applications with large DFAs.

Table 1. A Comparison of Complexities

	Round Complexity	client Computations		server Computations		Communication Complexity
		Asymmetric	Symmetric	Asymmetric	Symmetric	
Troncoso [15]	$O(n)$	$O(n\|Q\|)$	None	$O(n\|Q\|)$	$O(n\|Q\|)$	$O(n\|Q\|k)$
Frikken [2]	2	$O(n+\|Q\|)$	$O(n\|Q\|)$	$O(n+\|Q\|)$	$O(n\|Q\|)$	$O(n\|Q\|k)$
Gennaro [3]	$min(O(\|Q\|), O(n))$	$O(n\|Q\|)$	None	$O(n\|Q\|)$	None	$O(n\|Q\|k)$
Yao's protocol [16]	1	$O(n)$	$O(n\|Q\|\log\|Q\|)$	$O(n)$	$O(n\|Q\|\log\|Q\|)$	$O(n\|Q\|k)$
Ishai [8]	1	$O(n)$	None	$O(n\|Q\|)$	None	$O(kn^2)$
Protocol 1(PRG)	1	$O(n)$	$O(n)$	$O(n)$	$O(n\|Q\|)$	$O(n\|Q\|k)$
Protocol 1 (PRG+Extended OT)	1.5	$O(k)$	$O(n)$	$O(k)$	$O(n\|Q\|)$	$O(n\|Q\|k)$

5 Counting Accepting States and Secure Pattern Matching

Modified versions of our proposed protocol for Oblivious DFA evaluation can be used to efficiently solve multiple variants of the *Secure Pattern Matching* problem. This problem has been the focus of several recent works (e.g. see [3,6,9]). In this section we use the notion of Alice/Bob in which Alice has the role of the

server and Bob has the role of the client in our protocol. This notion helps us to better explain the secure pattern matching application. In the three main variants we consider here, one party (Bob) holds a text T while the other party (Alice) holds a pattern p and the aim is for Alice to learn one of the following: (i) whether or not p appears in T, (ii) all the locations (if any) where p occurs as a pattern in T, or (iii) the number of occurrences of a pattern p in T, while Bob learns nothing about the pattern.

The first two variants can be instantiated through a relatively straightforward application of our ODFA protocol from Section 4. Nevertheless, a few small modifications and considerations are necessary to make things work and we discuss them in the full version of the paper [12].

The more interesting and challenging problem to tackle is the third variant of secure pattern matching where parties are interested in counting the number of occurrences of the pattern in a text but nothing else. Counting the number of occurrences is a natural measure of how related or essential a pattern is to a studied text. While solving the second variant of the problem would also provide the number of occurrences of the pattern, it reveals significantly more information than just the count. Hence, if the number of occurrences is all that the parties are interested in, a solution for the second variant is not a suitable solution.

It is not clear how to modify existing secure pattern matching protocols to solve the third variant without a significant increase in their computation. It is also not clear how to formulate this problem as an oblivious DFA evaluation protocol and then apply our construction from Section 4 to it. We observe that what is really needed to solve this variant of the secure pattern matching problem, is a modified oblivious DFA evaluation protocol that counts the number of accepting states visited during an input evaluation and outputs this count as the final result as opposed to a single bit indicating whether the final state was an accepting or a rejecting one. This modified version of the ODFA protocol, when applied to the pattern-specific DFA of KMP [10], yields exactly a secure protocol for the third variant of the pattern matching problem. Our solution for this variant uses a novel trick (see Section 5.1 for details) that allows Alice to learn the number of occurrences of the pattern p without having to perform any additional computation.

5.1 Third Variant: Number of Locations of p in T

Now consider the more challenging variant where the goal is to only reveal the number of occurrences to Alice or Bob but nothing else. First consider the case where Bob is to learn the number of matches while Alice learns nothing. The pattern-specific DFA we need is again generated using the KMP algorithm [10]. The main observation is that for the KMP-transformed DFA, the number of accepting states visited in one evaluation of a text T, is exactly the number of times a pattern p occurs in a text T. Hence, all we need to do is to design a protocol for counting the the number of accepting states visited during a

DFA evaluation of the input. Such an oblivious DFA protocol might find other applications in future.

Modifications. Alice generates n uniformly random values $s_i \in F$ for $1 \leq i \leq n$ where F is a finite field of size $|F| > |T|$. Alice then computes $S = \sum_{i=0}^{n} s_i$. When generating the DFA matrix, for each row i of M_Γ, Alice concatenates the values in each cell by s_i except for the cells corresponding to accepting states for which the value s_i+1 is concatenated instead. The DFA matrix is then garbled as usual. Alice sends S along with the garbled DFA matrix to Bob. When computing the final output, Bob collects all the values $s'_i \in F$ for $1 \leq i \leq n$ on his transit path. Finally, Bob computes the sum of those values ($S' = \sum_{i=0}^{n} s'_i$), and outputs $S' - S$ as the number of occurrences of p in T.

Now if we only want Alice to learn the result, we do not send the value S to Bob. Instead, in the above protocol when Bob calculates S', he sends it back to Alice who computes $S' - S$ on his own in order to learn the number of matches.

Correctness. The intuition behind the correctness of the algorithm is that for each location i where p appears in T, the value $s_i + 1$ is retrieved by Bob and for all other locations the value s_i. Hence, the number of additional 1s is exactly equal to the number of locations of p in T.

Security. In order to prove the security of the scheme, we need to show that Bob cannot distinguish between s_i and $s_i + 1$ values he retrieves since they both are uniformly random values in F. In other words, we need to show that his view is simulatable given just the final output.

The proof of security in this case is slightly more subtle, since it does not automatically follow from our original ODFA protocol. Hence, we outline the intuition behind it next. Our main observation is that the following two distributions (D_V and D'_V) are *identically* distributed.

Let V be an arbitrary subset of size t of $\{1, \ldots, n\}$. Here, V represents the locations in T where matches occur. Consider the following two distributions:

1. (D_V) Choose n uniformly random values $\{s_1, \ldots s_n\} \in F$. Compute $S = \sum s_i$. For every i in V, let $s'_i = s_i + 1$. For the rest, let $s'_i = s_i$. Output $(s'_1, \ldots, s'_n)$.
2. (D'_V) Choose a uniformly random value S in F. Let $S' = S + t$. Generate $n-1$ random values $s'_1, \ldots, s'_{n-1}$. Let $s'_n = S' - \sum_{i=0}^{n-1} s'_i$. Output $(s'_1, \ldots, s'_n)$.

It is relatively easy to show that the above two distributions D_V and D'_V are identical for any subset $V \subset \{1, \ldots, n\}$ of size t. Given this property, we can modify the original proof of security for our oblivious DFA evaluation protocol such that the simulator in the proof of Theorem 1 samples from the second distribution (D'_V) while the first distribution (D_V) represents the distribution of the corrupted party's view in the real world execution of the protocol. Since sampling from D'_V only requires knowledge of t (i.e. the number of occurrences of p in T), our simulator can simulate the real world adversaries's view given only the final output. A complete proof of security for the above protocol closely follows the proof of Theorem 1, and hence is omitted.

6 Implementation and Experimental Results

To demonstrate that our proposed protocol as well as its variants are practical, we have implemented and evaluated our protocol. Implementation is done using C++ and the Crypto++ library v.5.61. The experiments were run on two machines one as the client (input holder) and the other as the server (DFA holder). Each of these systems has an Intel Core i7 processor, with 4GBs of RAM. They systems are connected using a 1 GB ethernet.

6.1 OT Implementation

For our OT protocol, we have implemented the Naor-Pinkas amortized OT (See section 3.1 of [13]) which requires one exponentiation for each transfer. We implemented their protocol over Elliptic Curves (EC) for better efficiency. The EC curve we use is the NIST recommended curve P-192 (see Section D.1.2.1 of [1]).

We have also implemented the OT extensions of [7] for improved efficiency. Two extensions are discussed in [7]. The first one is concerned with extending the number of OTs efficiently (Section 3) while the second extension (Appendix B [7]) reduces oblivious transfer for long strings to oblivious transfer of shorter strings.

Both extensions mentioned above rely on the use of a hash function (in the random oracle model). We have chosen SHA-256 for this implementation. When the number of OT invocations is lower than $k = 80$, we make a direct call to our base OT protocol, but otherwise employ the first extension to reduce the number of OT invocations. When we encounter an OT with message sizes larger than 256 bits (equivalently 32 bytes) we reduce them to an OT with message size of 256 bits using the second extension. Note that since in this protocol the message for each OT is XORed with the output of the random oracle (hash function), we are able to handle varying message sizes for each OT by simply adjusting the output size of the random oracle to the corresponding message size.

The PRG is also implemented using sufficient invocations of the random oracle (i.e. SHA-256).

Table 2. Empirical Results: running times in (ms)

n	Client Ungarbling	Client OT	Server Garbling	Server OT	Comm. (MB)
100	0.13	67.10	6.85	127.60	0.06
500	0.61	69.65	34.04	130.01	0.28
1500	1.92	74.40	102.22	137.20	0.83
5000	6.32	92.05	340.10	159.89	2.72
10000	12.66	118.36	674.59	191.05	5.44
20000	25.43	166.50	1352.59	254.48	10.88
50000	64.35	323.48	3409.71	448.51	27.19
75000	96.44	451.53	5056.24	610.65	40.78
100000	128.89	576.53	6794.88	782.05	54.38
150000	194.66	837.60	10244.40	1087.57	81.55

(a) Experiment 1

Q	Client Ungarbling	Client OT	Server Garbling	Server OT	Comm. (MB)
100	0.02	32.14	6.89	17.76	0.05
500	0.02	31.80	33.37	18.54	0.27
1500	0.02	31.90	99.74	18.26	0.80
5000	0.03	32.36	331.47	18.44	2.67
10000	0.03	31.86	666.18	18.51	5.34
20000	0.03	31.56	1332.67	18.39	10.68
50000	0.03	32.12	3373.53	17.92	26.71
75000	0.03	32.22	5169.60	18.88	42.92
100000	0.03	32.57	6949.88	18.56	57.22
150000	0.03	32.29	10640.30	18.44	85.83

(b) Experiment 2

6.2 Experiments

We have designed two experiments to analyze the effect of the input size and the DFA size on the performance of our protocol. In the first experiment we fix the DFA size and increase the input size, while in the second experiment we fix the input size and increase the DFA size. In what appears next, we refer to the input holder as the client while referring to the DFA holder as the server.

Experiment 1. In the first experiment, an arbitrary DFA with 20 states is considered. We have chosen a low number of states in order to draw a clear conclusion on the effect of the input size. We then increase the input size starting from 10 bits all the way up to 150000 bits. This experiment is of interest for applications such as DNA matching where the input can be large while the number of DFA states (related to the pattern size) is often low. It is noteworthy to mention that by fixitng the state size, the DFA transitions does not have any effect on computation or communication costs and hence we just selected a DFA with arbitrary transitions. The results of this experiment are presented in Table 2(a).

From Table 2(a), it can be observed that the client time is dominated by the client's OT time, and the client's evaluation (ungarbling) time is almost negligible for even large input sizes. This is due to the fact that the client's ungarbling is limited to only one PRG evaluation per input bit. On the other hand, based on Table 2(a), for large input sizes, server time is dominated by the server's garbling time. The reason for this is partly due to our use of OT extension, which prevents the number of exponentiations from increasing as the input size grows. We also note that that for input size of 2000 bits or more, OT time is no longer the bottleneck for the overall protocol. Furthermore, the server's garbling time is dependent on the size of the DFA matrix (unlike client's evaluation time which only depends on the number of rows of the matrix) and hence grows as we increase the input size.

Experiment 2. In the second experiment, for a fixed input size of 20 bits, we produce arbitrary DFAs with increasing number of states (10 to 150000 states). The results of this experiment are presented in Table 2(b).

Based on Table 2(b) we note that the server time is dominated by the server garbling time, since the number of OTs remain the same. The OT time for 20-bit inputs is approximately 32 milliseconds for both the client and the server. Client's ungarbling time is negligible because it does not depend on the number of states. When the input size is 20, for DFA sizes of over 200, the OT is no longer the bottleneck. Again we have a negligible computation time for the client and a total time (client + server) of under 1 second for DFAs with number of states as large as 15000.

Finally, we note that the communication time only constituted a small portion of the total time in our experiments and hence we only report the size of communication in the tables.

References

1. FIPS, P.: 186-3. Digital signature standard (DSS) (2009)
2. Frikken, K.: Practical Private DNA String Searching and Matching through Efficient Oblivious Automata Evaluation. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security 2009. LNCS, vol. 5645, pp. 81–94. Springer, Heidelberg (2009)
3. Gennaro, R., Hazay, C., Sorensen, J.: Text Search Protocols with Simulation Based Security. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 332–350. Springer, Heidelberg (2010)
4. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 169–178. ACM (2009)
5. Hazay, C., Lindell, Y.: Efficient Protocols for Set Intersection and Pattern Matching with Security against Malicious and Covert Adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
6. Hazay, C., Toft, T.: Computationally Secure Pattern Matching in the Presence of Malicious Adversaries. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 195–212. Springer, Heidelberg (2010)
7. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
8. Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
9. Katz, J., Malka, L.: Secure text processing with applications to private DNA matching. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 485–492. ACM (2010)
10. Knuth, D., Morris Jr, J., Pratt, V.: Fast pattern matching in strings. SIAM Journal on Computing 6, 323 (1977)
11. Lindell, Y., Pinkas, B.: A proof of Yao's protocol for secure two-party computation. Journal of Cryptology 22(2), 161–188 (2009)
12. Mohassel, P., Niksefat, S., Sadeghian, S., Sadeghiyan, B.: An efficient protocol for oblivious DFA evaluation and applications. Cryptology ePrint Archive, Report 2011/434 (2011), http://eprint.iacr.org/
13. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2001, pp. 448–457 (2001)
14. Sipser, M.: Introduction to the Theory of Computation. International Thomson Publishing (1996)
15. Troncoso-Pastoriza, J., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient dna searching through oblivious automata. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 519–528. ACM (2007)
16. Yao, A.: Protocols for secure computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. Citeseer (1982)

Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces

Seung Geol Choi[1], Kyung-Wook Hwang[2], Jonathan Katz[1], Tal Malkin[2], and Dan Rubenstein[2]

[1] University of Maryland
{sgchoi,jkatz}@cs.umd.edu
[2] Columbia University
{kwhwang@ee,tal@cs,danr@cs}.columbia.edu

Abstract. Protocols for generic secure multi-party computation (MPC) generally come in two forms: they either represent the function being computed as a *boolean* circuit, or as an *arithmetic* circuit over a large field. Either type of protocol can be used for any function, but the choice of which protocol to use can have a significant impact on efficiency. The magnitude of the effect, however, has never been quantified.

With this in mind, we implement the MPC protocol of Goldreich, Micali, and Wigderson [13], which uses a boolean representation and is secure against a semi-honest adversary corrupting any number of parties. We then consider applications of secure MPC in *on-line marketplaces*, where customers select resources advertised by providers and it is desired to ensure privacy to the extent possible. Problems here are more naturally formulated in terms of boolean circuits, and we study the performance of our MPC implementation relative to existing ones that use an arithmetic-circuit representation. Our protocol easily handles tens of customers/providers and thousands of resources, and outperforms existing implementations including FairplayMP [3], VIFF [11], and SEPIA [7].

1 Introduction

Protocols for secure multi-party computation allow a set of parties $P_1, \ldots, P_n$ to compute some function of their inputs in a distributed fashion, while revealing nothing to a coalition of corrupted parties about any honest party's input (or any group of honest parties' inputs), beyond what is implied by the output. Seminal results in cryptography dating to the 1980s [30,13,12] show that *any* polynomial-time function can be computed securely in the presence of coalitions of up to $n-1$ corrupted parties. For many years, the perception was that these were to be viewed as purely theoretical results with little practical relevance. This changed (to some extent) with the advent of Fairplay [24], an implementation of Yao's protocol for secure two-party computation that demonstrated for the first time that generic protocols were within the realm of feasibility. Since then, several implementations of generic secure two-party and multi-party computation

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 416–432, 2012.

protocols have been developed [3,22,5,11,26,7,15,23], and this is currently an active area of research.

In this work our focus is on generic protocols for secure *multi-party* computation (MPC) in the *semi-honest* setting. (In the semi-honest setting, parties are assumed to follow the protocol but coalitions of malicious parties may attempt to learn additional information from the joint transcript of their execution of the protocol. By "generic" we mean protocols that can be used to securely compute arbitrary functions.) There are, broadly speaking, two approaches taken by protocols for secure MPC: they either represent the function being computed as a *boolean* circuit, or as an *arithmetic* circuit over a (cryptographically) large field[1] $\mathbb{F}$. Although any function f can be computed using either type of protocol, the choice of representation affects the size of the circuit implementing f, and hence the overall efficiency of a secure protocol for computing f. The magnitude of the effect, however, has never been measured experimentally.

Most existing implementations of secure MPC rely on an arithmetic-circuit representation, with ShareMind [4], VIFF [11], and SEPIA [7] serving as prominent examples. We are aware of only one existing implementation of secure MPC (namely, FairplayMP [3]) using boolean circuits. As we will see, for certain problems a boolean-circuit representation is more natural, and so it is important to have protocols of both types available. Indeed, the motivation for our work came from trying to apply secure MPC to privacy-preserving computation in *on-line marketplaces*, where customers select resources advertised by providers and it is desired to ensure privacy to the extent possible. (See the following section for details.) In doing so, we found that existing implementations of secure MPC were unsuitable or too inefficient for our purposes. Moreover, all the MPC implementations mentioned above assume an *honest majority*, even though resilience against an arbitrary number of corruptions is known to be attainable.

1.1 Our Contributions

We implemented the classical MPC protocol of Goldreich, Micali, and Wigderson [13] (the *GMW protocol*), which uses a boolean-circuit representation for the function being computed and is secure against a semi-honest adversary controlling any number of corrupted parties. In our implementation, described in Section 2, we employ several optimizations to improve efficiency. Our code is publicly available[2] and we expect that, as with other systems, it will be useful in future work on privacy-preserving distributed computation.

With our system in place, any privacy-preserving multi-party computation can be solved, in principle, by defining an appropriate circuit for the task at hand. We designed circuits addressing three different (but related) problems in the context of on-line marketplaces where, generally speaking, *providers* advertise *resources* to be selected and subsequently utilized by *customers*, and the purpose of the

[1] Of course, a boolean circuit can be viewed as a circuit over the field $\mathbb{F} = GF(2)$. The distinction is that protocols using arithmetic circuits require $1/|\mathbb{F}|$ to be negligible in order for security and correctness to hold.

[2] `http://www.ee.columbia.edu/~kwhwang/projects/gmw.html`

marketplace is to match customers with providers in a way that optimizes some value under a certain set of constraints. We look at the following examples:

- **P2P Content-Distribution Services.** [9,18] provide a marketplace where *content* is the resource, and providers advertise availability of content at peers. Here, a customer may want to determine which peer hosting the desired content is the best choice (e.g., closest, or having minimal delay) for retrieving that content.
- In **Cloud Computing.** providers are cloud platforms (e.g., Amazon EC2, Microsoft Azure, etc.), resources are the services (e.g., storage, bandwidth, or processing) offered by each provider, and customers want to find the provider(s) offering services matching their needs at the cheapest price [1,8, 29,27,28,20].
- A **Mobile Social Network.** can be viewed as a marketplace where users are both customers and resources, and the provider helps users locate and connect to other users who share similar interests.

Formal definitions of the problems in each of the above settings are given in Section 3, and we describe optimized circuits solving each of them in the full version of this paper [10]. For these problems, we find that it significantly helps to be able to work with boolean circuits rather than arithmetic circuits.

In Section 4 we evaluate the performance of our MPC protocol as applied to one of the above problems. (Since they have similar circuits, the other two problems should display similar results.) Our results shows that our protocol can be used to efficiently and securely implement a distributed marketplace with tens of providers/customers and thousands of resources over a wide-area network. Our implementation outperforms systems such as VIFF [11] and SEPIA [7], in part because we use boolean circuits rather than arithmetic circuits as those systems do.[3] Another advantage of our protocol is that it provides security against *any number* of corruptions, whereas the cited implementations [3,4,11,7] all require an honest majority.

1.2 Other Related Work

There are several existing implementations of secure two-party computation [24, 22,26,14,15,23]. These are all specific to the two-party setting and do not yield protocols for three or more parties. Interestingly, and in contrast to other multi-party implementations [3,11,7] that *only* handle three or more parties, the GMW protocol we implement can handle any number of parties $n \geq 2$. For the two-party case, however, we expect our implementation to be roughly a factor of two slower than the best available system [15].

Other implementations of secure multi-party computation, besides those discussed above, include [6,4,17]. The code for SIMAP [6] is not publicly available,

[3] FairplayMP [3] also uses boolean circuits, but did not support multiple input values per party and crashed on the input sizes used. See Section 4 for further discussion. We do not compare to ShareMind since that system only supports 3-party computation.

and anyway SIMAP appears to be superseded by VIFF. Sharemind [4] handles *only* the three-party setting, assuming at most one semi-honest corruption. The work of Jakobsen et al. [17] achieves resilience against an arbitrary number of *malicious* corruptions. Their implementation is based on arithmetic circuits and has worse performance than VIFF (though with better resilience).

2 MPC Implementation

We provide an overview of the GMW protocol and details of our implementation. The GMW protocol provides security against a semi-honest adversary corrupting any number of parties. (We refer to [12] for formal definitions of security.) Assuming semi-honest behavior is reasonable in settings where the codebase is difficult to change without detection, where software attestation can be used to convince other parties that correct software is being run, or where parties are trusted but must ensure secrecy of data for policy reasons or because of concerns about future break-ins.

2.1 Overview of the GMW Protocol

1-out-of-4 Oblivious Transfer. *Oblivious transfer* (OT) is a key building block of the GMW protocol. A 1-out-of-4 OT protocol is a two-party protocol in which there is a sender holding values (x_0, x_1, x_2, x_3) and a receiver holding an index $i \in \{0, \ldots, 3\}$; the receiver learns x_i, but neither the sender nor the receiver learn anything else; i.e., the receiver learns nothing about any other values held by the sender, and the sender learns nothing about the receiver's index.

Details of our OT implementation are given in Section 2.2.

The GMW Protocol. The GMW protocol assumes the function f to be computed is represented as a boolean circuit consisting of XOR and AND gates or, equivalently, gates for addition and multiplication modulo 2. Let n denote the number of parties. In the GMW protocol the parties maintain random n-out-of-n shares $(s_{w1}, \ldots, s_{wn})$ of the value s_w on each wire w in the circuit; that is, party P_i holds share s_{wi} and all shares are random subject to $s_w = \bigoplus_i s_{wi}$. Setting up such shares on the input wires is easy: party P_i with input s_w on wire w chooses random s_{wj} for $j \neq i$, sends s_{wj} to P_j, and locally sets $s_{wi} = s_w \oplus \left(\bigoplus_{j \neq i} s_{wj}\right)$. Shares on internal wires of the circuit are then computed inductively in the following way:

XOR gates. Say w is the output wire of an XOR gate with input wires u and v, and the parties have shares $(s_{u1}, \ldots, s_{un})$ and $(s_{v1}, \ldots, s_{vn})$ of s_u and s_v, respectively. Then each party P_i locally computes $s_{wi} = s_{ui} \oplus s_{vi}$, and one can observe that $(s_{w1}, \ldots, s_{wn})$ is a valid sharing of $s_w = s_u \oplus s_v$.

AND gates (cf. [12]). Say w is the output wire of an AND gate with input wires u and v, and the parties have shares $(s_{u1}, \ldots, s_{un})$ and $(s_{v1}, \ldots, s_{vn})$ of s_u and s_v, respectively. Note that

$$s_w = s_u \cdot s_v = \left(\sum_{i=1}^{n} s_{ui}\right) \cdot \left(\sum_{i=1}^{n} s_{vi}\right) = \sum_{i=1}^{n} s_{ui}s_{vi} + \sum_{i<j} (s_{ui}s_{vj} + s_{uj}s_{vi}).$$

Each party P_i can compute $s_{ui}s_{vi}$ locally. As for the remaining term, each pair of parties P_i, P_j computes a random additive share of $s_{ui}s_{vj} + s_{uj}s_{vi}$ in the following way. P_j chooses a random bit $c_j^{\{i,j\}}$, and computes four values

$$c_j^{\{i,j\}}, \quad c_j^{\{i,j\}} \oplus s_{vj}, \quad c_j^{\{i,j\}} \oplus s_{uj}, \quad c_j^{\{i,j\}} \oplus s_{vj} \oplus s_{uj}$$

corresponding to the four possible values of P_i's shares s_{ui}, s_{vi}. Party P_i then acts as a receiver in 1-out-of-4 OT, with index determined by the actual values of its shares s_{ui}, s_{vi}, to obtain the appropriate value from P_j that we denote by $c_i^{\{i,j\}}$. Note that $c_i^{\{i,j\}} + c_j^{\{i,j\}} = (s_{ui}s_{vj} + s_{uj}s_{vi})$. Finally, each party P_i computes

$$s_{wi} = s_{ui}s_{vi} + \sum_{j \neq i} c_i^{\{i,j\}}.$$

It can be verified that $(s_{w1}, \ldots, s_{wn})$ is a (random) sharing of $s_w = s_u \cdot s_v$.

Evaluation of XOR gates is essentially free, whereas evaluating AND gates requires $\binom{n}{2}$ invocations of 1-out-of-4 oblivious transfer.

Once a sharing $(s_{w1}, \ldots, s_{wn})$ of an output wire w is obtained, the value s_w can be reconstructed by having each party privately send its share to all other parties. It is also possible for only some specific party to learn a given output value by sending shares to that party only. We note that this is the only step in the protocol where private channels are needed, and then only if more than one party is to learn a given output value.

2.2 Oblivious-Transfer Protocols

As noted in the previous section, oblivious transfer is a key building block of the GMW protocol; it is also the most computationally expensive part of the protocol, since it is the only part of the protocol that relies on public-key techniques. As described above, the GMW protocol requires one invocation of 1-out-of-4 OT per pair of parties each time an evaluation of an AND gate is performed, and so m executions of OT (per pair of parties) to evaluate a circuit containing m AND gates. We can improve the overall efficiency, however, using two techniques:

- Using *OT pre-processing* [2], each pair of parties can perform m oblivious transfers *on random inputs* at the outset of the protocol, and then (very efficiently) use the pre-computed values thus obtained to achieve the functionality of oblivious transfer on their actual inputs when evaluating an AND gate. Thus, all the oblivious transfers that will be needed throughout the entire protocol can be run *in parallel* at the beginning of the protocol.

k parallel invocations of 1-out-of-4 OT

Let $g, \mathbb{G}$, and q be fixed, where $\mathbb{G}$ is a cyclic group of prime order q, and g is a generator of $\mathbb{G}$. Let $H : \{0,1\}^* \to \{0,1\}^m$ be a hash function.

INPUTS. **S** holds $\{(x_0^j, x_1^j, x_2^j, x_3^j)\}_{j\in[k]}$ with $x_i^j \in \{0,1\}^m$. **R** holds $(r_1, \ldots, r_k)$ where $r_j \in \{0, \ldots, 3\}$.

THE PROTOCOL.

1. **S** chooses $\alpha \leftarrow \mathbb{Z}_q$ and computes $c_0 = g^\alpha$, and also chooses $c_1, c_2, c_3 \leftarrow \mathbb{G}$. It sends $c_0, \ldots, c_3$ to **R**.

For $j \in [k]$ the parties do:

2. **R** chooses $\beta_j \leftarrow \mathbb{Z}_q$. If $r_j = 0$ then it sets $d_j = g^{\beta_j}$; else, it sets $d_j = c_{r_j}/g^{\beta_j}$. Finally, **R** sends d_j to **S**.
3. **S** computes $e_0 = d_j^\alpha$ and $e_i = (c_i/d_j)^\alpha$ for $i \in \{1,2,3\}$. Then **S** sends $\bar{x}_i^j = H(e_i, j, i) \oplus x_i^j$ to **R** for $i \in \{0, \ldots, 3\}$.
4. **R** computes $c_0^{\beta_j} = e_{r_j}$, and then outputs $x_{r_j}^j = \bar{x}_{r_j}^j \oplus H(e_{r_j}, j, r_j)$.

Fig. 1. The Naor-Pinkas OT protocol

– Using *OT extension* [16,21], it is possible to achieve the functionality of m invocations of 1-out-of-4 OT at essentially the cost of k invocations of 1-out-of-4 OT of m-bit strings, where k is a statistical security parameter. (More precisely, the marginal cost for each additional OT is just a small number of hash computations.) Security here is based on the assumption that the hash function is *correlation robust* [16].

Combining these optimizations, each pair of parties needs only run k (parallel) invocations of some "base" OT protocol (for m-bit strings) at the outset. These can be converted to $m \gg k$ OT executions (on bits) using OT extension; these m "pre-processed" OTs can then be used, as needed, during the rest of the protocol. It remains only to specify the "base" 1-out-of-4 OT protocol we use. We take as our base OT protocol the one by Naor and Pinkas [25], secure under the decisional Diffie-Hellman (DDH) assumption in the random-oracle model. Their protocol (actually, a version implementing k parallel executions of their protocol) is described in Figure 1 for completeness.

2.3 Implementation Details

We implemented the GMW protocol in C++. Our implementation takes as input a file containing a description of a boolean circuit for the function f of interest. (All parties are assumed to be running with identical copies of the circuit.) See Section 2.4 for an example. Unlike FairplayMP [3], we do not provide a mechanism for compiling a high-level language into a boolean circuit.

Parallelism. Nowadays, it is common for computers to have multiple cores. We use multi-threaded programming so as to take advantage of the available parallelism. In particular, each OT execution is performed by a separate thread. In the OT extension protocol, we optimize execution time by having parties send values as soon as they are computed, rather than waiting for the other party to

finish sending. (This does not affect security, since this occurs at fixed times that are independent of the parties' inputs and we assume semi-honest behavior.)

Random Oracle. We use SHA-1 to implement a random oracle H with arbitrary output length by defining

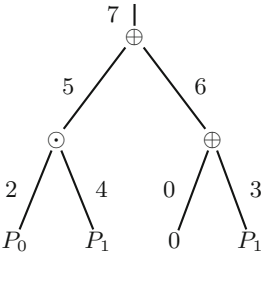
$$H(M) = \text{SHA-1}(\mathsf{seed}, 0)||\text{SHA-1}(\mathsf{seed}, 1)||\cdots,$$

where $\mathsf{seed} = \text{SHA-1}(M)$. Note that seed need only be computed once. We use the SHA-1 implementation of PolarSSL (`http://polarssl.org`).

Oblivious Transfer. For our base OT protocol we use the Naor-Pinkas protocol (see Figure 1) with group $\mathbb{G} \subset \mathbb{Z}_p$ of prime order q, and $p = 2q + 1$ with p prime. In our default implementation, p is a 1024-bit integer. We modified the modular-arithmetic module of NTL (`http://www.shoup.net/ntl`) to be thread-safe, and used it to implement the base OT protocol. Recall we use OT extension to improve efficiency. By default, we use statistical security parameter $k = 80$ in our implementation. Messages are transmitted in chunks of reasonable size to obtain a balance between the idle time and the number of socket calls.

2.4 Circuit Example

Our implementation of the GMW protocol takes as input (at each party running the protocol) three files that contain configuration information, the input of the party in question, and a description of a boolean circuit for the function f of interest. (All parties are assumed to be running with identical copies of the circuit.) In Figure 2 we show an example circuit along with its description using our representation. The circuit description uses the following format:




```
n 2
d 7 5 2
i 0 2 2
i 1 3 4
o 0 1 0
o 1 7 7
v 0 1
v 1 1
g 0 0 -1 -1 0
g 1 0 -1 -1 0
g 2 0 -1 -1 1 5
g 3 0 -1 -1 1 6
g 4 0 -1 -1 1 5
g 5 1 2 4 1 7
g 6 2 0 3 1 7
g 7 2 5 6 0
```

Fig. 2. Circuit Example

- The first line of the file has the form `n` X, where X denotes the number of parties participating in the protocol.
- The second line of the file contains a `d` followed by the total number of wires in the circuit, the number w of the first non-input wire (i.e., wires 0 to $w-1$ are input wires), and the number of XOR gates in the circuit.
- For each party, there is a line containing an `i` followed by the party's id, the number of the first input wire belonging to that party, and the number of the last input wire belonging to that party. (We assume wires are numbered such that every party provides inputs on a consecutive set of wires.)
- For each party, there is a line in the file containing an `o` followed by the party's id, the number of the first output wire belonging to that party, and

the number of the last output wire belonging to that party. (We assume wires are numbered such that every party receives outputs on a consecutive set of wires.) If a party receives no output, the number of the last output wire for that party is set to 0.
- For each party, there is a line in the file containing a `v` followed by the party's id and then an integer denoting the number of bits that should be used to represent each item in that party's input file. (E.g., if the input file of party 1 contains a '4' then this value will be represented as the 3-bit integer '100' if this line of the file is '`v 1 3`', but will be represented as the 5-bit integer '00100' if this line of the file is '`v 1 5`'.) Each bit in the ultimate representation of the integer will correspond to one of the input wires of the party.
- The remaining lines of the file describe the gates in the circuit. For each gate, we list (a) the number of the output wire of this gate (which also serves as the gate id); (b) the gate type, which can be either input (0), AND (1), or XOR (2); (c) the numbers of the left and right input wires (set to -1 if these are input gates); and (d) the out-degree of the gate. If the out-degree is non-zero, then the ids of the gates that receive the output of the current gate are listed. Gate ids 0 and 1 are reserved for the constants 0 and 1, respectively.

3 Problem Definitions

We introduce three problems in the context of on-line marketplaces where, generally speaking, *providers* advertise *resources* to be selected and subsequently utilized by *customers*, and the function of the marketplace is to match customers with providers so as to optimize some value under a certain set of constraints. As highlighted in the Introduction, we look at examples in the settings of P2P content distribution, cloud computing, and mobile social networks.

As a toy example, consider a customer who wishes to buy a car (resource) from one of several dealers (providers). The customer is interested in several different models of cars (but not all models); the different providers offer a variety of models (not all of which interest the customer); and each provider prices each model independently. The customer wishes to find an acceptable car at the lowest cost, without revealing the set of models he or she is interested in; the providers do not want to reveal their prices. Secure MPC allows the customer to learn the identity of a provider selling an acceptable model at the lowest price, with the customer learning no other prices (or which models are sold by each provider), and with the providers learning only of the customer's willingness to buy some particular model at the given price.

More formally, let R be some set of resources. The input of each provider P_i is a collection of values for the resources in some subset $R_i \subseteq R$; i.e., P_i's input is of the form $\{v_r^i\}_{r \in R_i}$. (If desired, each P_i could just use some default value $v_r^i = \perp$ for $r \notin R_i$; in that case, we may simply write P_i's input as $\{v_r^i\}_{r \in R}$.) We look at marketplaces where the computation can be broken into the following two steps, which will be executed as a *single* secure computation (so only the final output is revealed, not the intermediate results after the first step):

1. For each provider P_i and resource $r \in R_i$, compute a *scoring function* $\mathsf{sc}_r^i = \mathsf{Score}(i, r, v_r^i, x_n)$, where x_n denotes the private input of the customer. (In the running toy example, each model is scored by its offered price if the model is of interest to the customer, and by ∞ otherwise.)
2. Next, apply a *best-match function* $\mathcal{B}$ to the set of sc_r^i values to obtain a result that is given to the customer. (In the toy example, $\mathcal{B}$ outputs (i, r, sc_r^i) with minimum sc_r^i.)

We allow the scoring function to be arbitrary. For the best-match function we consider two possibilities: either $\mathcal{B}$ returns a single (i, r) maximizing/minimizing sc_r^i (with ties broken arbitrarily, and with or without including sc_r^i as part of the output), or $\mathcal{B}$ returns the set of all (i, r) for which the score sc_r^i is greater/lower than some threshold. In the following subsections we instantiate this general framework in several specific scenarios.

3.1 P2P Content-Distribution Services

In our P2P content-distribution setting, content is replicated across various P2P servers or source peers (such as seeders) whose pairwise communications are measured (and perhaps even controlled) by network providers such as ISPs. Before a peer starts downloading content, he or she would like to find out the best source peer (with respect to network bandwidth, end-to-end delays, throughput, and so on) from which to receive the content.

Here the providers are the ISPs and the resources are the source peers themselves (which for simplicity we identify with their indices). Let R be the set of source peers, with $|R| = k$. We assume that the ISP to which each source peer is bound is public knowledge, so ISP P_i is associated with some set of peers R_i. The input of each ISP/provider P_i is the measured bandwidth v_r^i for each peer/resource $r \in R_i$. The customer knows which peers have a replica of the item it wishes to retrieve, and holds as secret input a vector $x_n = (b_1, \ldots, b_k)$ where $b_r = 1$ iff peer/resource r has the desired content, and $b_r = 0$ otherwise. The objective is for the customer to find the best (e.g., highest-bandwidth) peer among those holding the desired content, without revealing which source peers have the content; the ISPs do not want to reveal the bandwidth of their peers.

Here the scoring function can be defined as:

$$\mathsf{Score}\left(i, r, v_r^i, (b_1, \ldots, b_k)\right) = \begin{cases} v_r^i \text{ if } b_r = 1 \\ 0 \quad \text{otherwise} \end{cases},$$

and the best-match function $\mathcal{B}$ returns i, r maximizing sc_r^i. (In fact, it suffices to return r here since the provider to which r is bound is irrelevant and anyway known.)

3.2 Cloud Computing

In this setting, providers offer various service packages and the customer wants to select the service package meeting its needs at the lowest available price. The service packages offered by the providers are the resources here, and each such

resource r has a value $v_r = (q_r, p_r)$ that is composed of its service quality q_r and price p_r. (For simplicity, we treat service quality as a one-dimensional quantity, e.g., CPU cycles. Our treatment can easily be generalized.) The customer holds input (q, p), where q represents a minimum acceptable service quality and p is a maximum budget. Two scenarios can be considered: either the customer wants to find the cheapest resource r with $q_r \geq q$, or the highest-quality resource r with $p_r \leq p$; each of these cases is treated below. In either case, the customer never reveals its budget or its service requirements to any of the providers, nor do the providers reveal to the customer (or to each other) what service packages they are offering.

Lowest-Price Selection. In this formulation, the customer seeks the package that satisfies its requirements at the lowest price. Here we may define the scoring function as:

$$\mathsf{Score}\left((i, r, q_r^i, p_r^i),\ (q, p)\right) = \begin{cases} p_r^i \text{ if } q_r^i \geq q \text{ and } p_r^i \leq p \\ \infty \text{ otherwise} \end{cases}.$$

The best-match function $\mathcal{B}$ returns an i, r minimizing sc_r^i.

Highest-Quality Selection. Here the customer seeks the package that meets its budget while giving the highest quality service. Now we may define the scoring function as:

$$\mathsf{Score}\left((i, r, q_r^i, p_r^i),\ (q, p)\right) = \begin{cases} q_r^i \quad \text{if } q_r^i \geq q \text{ and } p_r^i \leq p \\ -\infty \text{ otherwise} \end{cases},$$

and the best-match function returns an i, r maximizing sc_r^i.

3.3 Mobile Social Networks

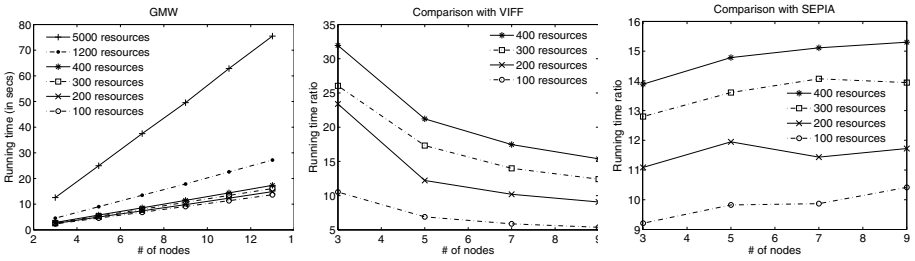
Here we consider a scenario where a user in a social network wants to identify nearby users who share common interests. Now the resources and providers are just the set R of all users (and the customer is one of the users as well), and the value of each "resource" (i.e., user) is that user's location and set of interests.

We assume that each user knows only about its own location and interests. Thus for each $r \in R$ we define $v_r^r = (\ell_r, H_r)$, where ℓ_r is the location of user r, and H_r is the set of that user's interests (perhaps represented as a bit-vector). The customer's input consists of (ℓ, H, δ) where ℓ is the location of the customer, H is the set of interests she wants a potential match to share, and δ is the distance radius in which she wants to search. We consider a few alternatives for what the customer wants as output.

Find all Close Matches. In this formulation, the customer wants to find all users within distance δ who share interests H. We may then define

$$\mathsf{Score}\left((r, \ell_r, H_r),\ (\ell, H, \delta)\right) = \begin{cases} 1 \text{ if } H \subseteq H_r \text{ and } |\ell_r - \ell| \leq \delta \\ 0 \text{ otherwise} \end{cases},$$

and the best-match function returns the set of all r such that $\mathsf{sc}_r^r = 1$.



(a) GMW (b) $\frac{\texttt{VIFF}}{\texttt{GMW}}$ (c) $\frac{\texttt{SEPIA}}{\texttt{GMW}}$

Fig. 3. Running times in a LAN

Find Closest Match. Here the customer wants to find the *closet* user who matches her interests. Now, define

$$\mathsf{Score}\left((r,\ell_r,H_r),\,(\ell,H,\delta)\right) = \begin{cases} |\ell_r-\ell| \text{ if } H\subseteq H_r \text{ and } |\ell_r-\ell|\le\delta \\ \infty \qquad\quad \text{otherwise} \end{cases}$$

The best-match function returns an r minimizing sc_r^r.

Find Best Resource. Now the customer would like to obtain the resource within radius δ that shares as many interests as possible. We thus define

$$\mathsf{Score}\left((\ell_r,H_r),\,(\ell,H,\delta)\right) = \begin{cases} |H_r\cap H| \text{ if } |\ell_r-\ell|\le\delta; \\ -\infty \qquad\quad \text{otherwise} \end{cases},$$

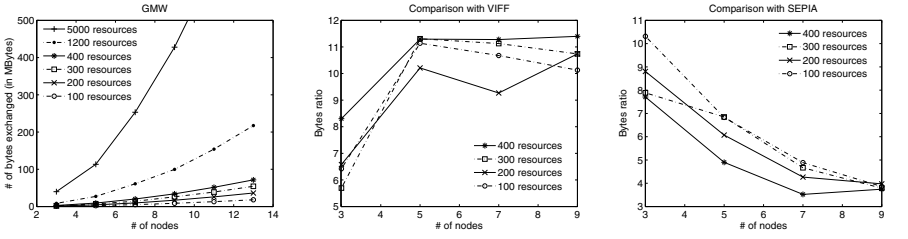
and the best-match function returns r maximizing sc_r^r.

3.4 Boolean-Circuit Constructions

We construct appropriate boolean circuits solving each of the problems described above. Since XOR gates are essentially "free" to evaluate in the GMW protocol, whereas evaluating each AND gate requires cryptographic computations, we aimed to minimize the number of AND gates in the circuits. Due to lack of space, descriptions of our circuits are given in the full version [10].

4 Performance Evaluation

We evaluate the performance of our implementation in both a local-area network (LAN) and a wide-area network (using PlanetLab, see `http://www.planet-lab.org/`), and compare it to existing systems for secure MPC. In our experiments we consider only the P2P content-distribution problem formulated in Section 3.1 with 16-bit integer representation (i.e., $\ell = 16$), but since circuits for the other two problems are similar (in terms of both circuit depth and the number of AND gates), we expect the results to be similar for those problems as well. We let `GMW` refer to our implemented protocol for this problem, obtained by applying our GMW implementation to the circuit (described in the full version [10]). All reported measurements are based on averages over 10 runs of the experiment in question.



(a) GMW (b) $\frac{\text{VIFF}}{\text{GMW}}$ (c) $\frac{\text{SEPIA}}{\text{GMW}}$

Fig. 4. Total bytes transferred among all nodes

4.1 Local-Area Network

Our first set of experiments is performed in a cluster consisting of multiple Linux host nodes, each containing two Intel Xeon 2.80GHz CPUs and 4GB RAM. We use one host per participant in the protocol, so an experiment with n providers involves an $(n+1)$-party multi-party computation on $n+1$ host machines. We set up our experiments so the customer chooses half the resources offered by each provider to be "of interest". Note that the client's inputs do not affect performance, since the same underlying circuit is evaluated regardless of the customer's input; indeed, if performance were affected by the customer's input then the protocol could not be secure.

We ran experiments using from 3 to 13 host nodes, and 100 to 5,000 resources. (This represents the aggregate offered by all providers.) For this problem, the number of AND gates being evaluated depends on the number of resources only (it is independent of the number of nodes), and ranges from about 5,500 AND gates (for 100 resources) to roughly 305,000 AND gates (for 5,000 resources). The running time is plotted in Figure 3(a), and the total bandwidth (between all parties) is shown in Figure 4(a).

For a fixed number of resources, the bandwidth grows quadratically with the number of nodes; this is because each pair of parties communicates for every AND gate being evaluated. The running time scales linearly with the number of nodes since all nodes work in parallel, and the work per node increases in direct proportion to the number of other nodes with which it communicates. Although difficult to see from the plots, for a fixed number of parties the running time and bandwidth increase roughly linearly in the number of resources k; this is because the size of the circuit grows roughly linearly in k (actually, it grows as $k \log k$ but the logarithmic term is difficult to detect).

We also measured the marginal time to evaluate a single AND gate (i.e., the time required to evaluate one additional AND gate, once the number of AND gates is large). We use marginal time because there is a fixed cost for the initial oblivious transfers performed by the parties, but then oblivious-transfer extension is used to get additional OTs at much lower cost (see Section 2.2). The marginal cost per AND gate ranged from 50 μs (for 3 parties) to 340 μs (for 13 parties).

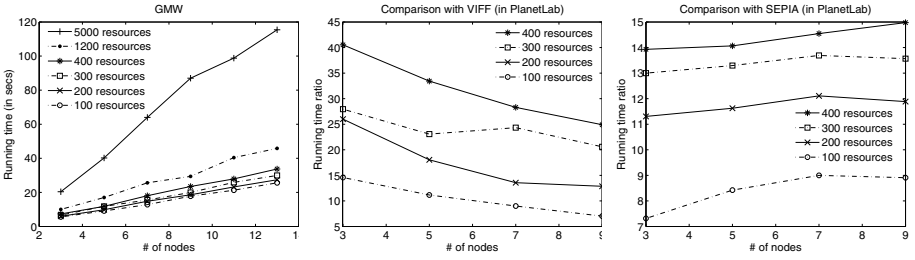
Comparison to Existing Work. We applied other existing implementations of secure MPC to the same problem. Unfortunately, despite contacting the authors we were unable to get a working implementation using FairplayMP [3] since we found that it did not support providing users with multiple inputs, and it would crash (when parties were provided with a single input) on inputs more than 16 bits long.[4] We were able to compare our protocol with implementations in (the semi-honest version of) VIFF [11] and SEPIA [7]. We ran both VIFF and SEPIA over insecure (i.e., non-SSL-protected) channels even though private channels would be needed to ensure security against an eavesdropping adversary for those protocols. (In contrast, for GMW a secure channel is not needed if only one party learns the output; when multiple parties learn the output, only the final-round messages need to be encrypted.) In SEPIA, parties provide their inputs to "privacy peers" that run a secure-computation protocol on their behalf; when we refer to "nodes" in SEPIA we mean the number of privacy peers.

In contrast to GMW, VIFF and SEPIA utilize arithmetic circuits where each wire carries an element of a large field $\mathbb{F}$ (with $\log |\mathbb{F}| \approx 64$ in each case), and gates perform addition or multiplication in $\mathbb{F}$. (Similar to the GMW case, addition is essentially "for free" whereas multiplication is "expensive".) The boolean circuit we used for GMW is easily adapted for VIFF/SEPIA as follows (see the full version of this paper [10] for further details):

- Boolean values can be represented as elements of $\mathbb{F}$. Boolean operations can be performed as $\mathrm{AND}(a,b) = ab$ and $\mathsf{XOR}(a,b) = a + b - 2ab$ (where computations are in $\mathbb{F}$), so long as $a, b \in \{0,1\}$. (Note, however, that both operations involve a multiplication in $\mathbb{F}$.)
- ℓ-bit integers can be represented as elements of $\mathbb{F}$, since $|\mathbb{F}| \gg 2^\ell$ for the value of ℓ we use. Because of this, addition and subtraction gates are now trivial to implement since they correspond exactly to addition and subtraction over $\mathbb{F}$.
- VIFF and SEPIA already provide circuits for performing comparisons.

In Figures 3(b) and 3(c) (resp., Figures 4(b) and 4(c)) we compare GMW's running time (resp., bandwidth utilization) to that of VIFF and SEPIA. Since the running times of VIFF and SEPIA are comparatively long, we only ran experiments with up to 400 resources and up to 9 nodes. In those ranges of the parameters, GMW completes in under 20 seconds while VIFF and SEPIA take an order of magnitude longer; the relative performance of GMW becomes even better as the number of resources is increased. The results demonstrate that our implementation scales significantly better, and is more efficient, than prior implementations. Recall also that GMW withstands a larger number of corruptions than either VIFF or SEPIA.

[4] We did compare the performance of GMW to FairplayMP for 5-party computation of a "toy" circuit consisting of a depth-d full binary tree of AND gates. With $d = 12$, GMW ran about 20 times faster than FairplayMP. FairplayMP crashes on circuits with $d > 12$, whereas GMW ran on circuits up to $d = 23$ (about 8 million gates).



(a) GMW (b) $\frac{\text{VIFF}}{\text{GMW}}$ (c) $\frac{\text{SEPIA}}{\text{GMW}}$

Fig. 5. Running times in PlanetLab

4.2 Wide-Area Network

We explored the effects of communication latency by running our implementation of the GME protocol in a wide-area network (WAN) using PlanetLab (`http://www.planet-lab.org/`). In the PlanetLab settings we explored, the maximum round trip time (RTT) was more than 200 ms. The test nodes in PlanetLab have various hardware specs; the least powerful node had two Intel Core2Duo 2.33GHz CPUs and 2.0GB memory, while the most powerful node had four Intel Xeon 2.83GHz CPUs and 3.7GB memory.

Figure 5 shows that GMW's running time increases by 17–64% relative to the time required on a LAN. (The bandwidth usage is identical whether running over a LAN or a WAN.) We also observed more variability in the running time over PlanetLab than in a LAN, which is not surprising. As we increase the number of participating nodes, the running time increases linearly (as in the LAN) even though nodes' configurations are not homogeneous; this suggests that performance is mostly affected by communication latency. GMW maintains stable performance regardless of network conditions and heterogeneous hardware configurations, consistently outperforming VIFF and SEPIA as shown in Figures 5(b) and 5(c).

5 Conclusions

We have shown an implementation of the GMW protocol for secure multi-party computation. Our implementation is distinguished from existing implementations of multi-party computation in two important ways: (1) Our implementation supports boolean circuits, rather than arithmetic circuits as in [11,7], and (2) it provides security against a semi-honest adversary corrupting any number of parties, rather than requiring an honest majority as in [3,11,7]. We have also shown that our implementation outperforms previous work [11,7], at least for certain classes of problems that are more amenable to being solved using boolean circuits rather than arithmetic circuits. Finally, our work shows the feasibility of applying generic secure multi-party computation to realistic networking problems where privacy is required.

Acknowledgments. Research of Jonathan Katz and Seung Geol Choi was supported by DARPA, and by NSF awards #0447075, #0964541, and #1111599. Research of Tal Malkin was supported by a Google research grant, NSF awards #0831094 and #1116702, and IARPA via DoI/NBC contract number D11PC20194. Research of Dan Rubenstein and Kyung-Wook Hwang was supported by NSF award #1017934 and DHS HSHQDC-10J00204. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation herein. The views and conclusions of this paper are those of the authors do not necessarily reflect the position or the policy of the US Government, DARPA, IARPA, or DoI/NBS, and no official endorsement should be inferred.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, UC Berkeley (2009)
2. Beaver, D.: Precomputing Oblivious Transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995)
3. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: A system for secure multi-party computation. In: 15th ACM Conf. on Computer and Communications Security, pp. 257–266. ACM Press (2008)
4. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A Framework for Fast Privacy-Preserving Computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008)
5. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Secure Multiparty Computation Goes Live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
6. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)
7. Burkhart, M., Strasser, M., Many, D., Dimitropoulos, X.: SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In: 19th USENIX Security Symposium, pp. 223–240. USENIX Association (2010)
8. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. Proc. IEEE 93(3), 698–714 (2005)
9. Chen, Y., Katz, R.H., Katz, Y.H., Kubiatowicz, J.D.: Dynamic Replica Placement for Scalable Content Delivery. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 306–318. Springer, Heidelberg (2002)
10. Choi, S.G., Hwang, K., Katz, J., Malkin, T., Rubenstein, D.: Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. Cryptology ePrint Archive, Report 2011/257 (2011), http://eprint.iacr.org/2011/257

11. Damgård, I., Geisler, M., Krøigaard, M., Nielsen, J.B.: Asynchronous Multiparty Computation: Theory and Implementation. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 160–179. Springer, Heidelberg (2009)
12. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
13. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: 19th ACM STOC Annual ACM Symposium on Theory of Computing (STOC), pp. 218–229. ACM Press (1987)
14. Henecka, W., Kögl, S., Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: TASTY: Tool for automating secure two-party computations. In: 17th ACM Conf. on Computer and Communications Security (CCCS), pp. 451–462. ACM Press (2010)
15. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: 20th USENIX Security Symposium. USENIX Association (2011)
16. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
17. Jakobsen, T.P., Makkes, M.X., Nielsen, J.D.: Efficient Implementation of the Orlandi Protocol. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 255–272. Springer, Heidelberg (2010)
18. Kangasharju, J., Roberts, J., Ross, K.W.: Object replication strategies in content distribution networks. Computer Communications 25(4), 376–383 (2002)
19. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009)
20. Lewis, P.R., Marrow, P., Yao, X.: Evolutionary Market Agents for Resource Allocation in Decentralised Systems. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 1071–1080. Springer, Heidelberg (2008)
21. Li, B., Li, H., Xu, G., Xu, H.: Efficient reduction of 1-out-of-n oblivious transfers in random oracle model. Cryptology ePrint Archive, Report 2005/279 (2005)
22. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing Two-Party Computation Efficiently with Security against Malicious Adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
23. Malka, L., Katz, J.: VMCrypt — modular software architecture for scalable secure computation, http://eprint.iacr.org/2010/584
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: 13th USENIX Security Symposium, pp. 287–302. USENIX Association (2004)
25. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology 18(1), 1–35 (2005)
26. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
27. Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: Trading grid services — a multi-attribute combinatorial approach. European J. Operational Research 187(3), 943–961 (2008)

28. Tan, Z., Gurd, J.R.: Market-based grid resource allocation using a stable continuous double auction. In: Proc. 8th IEEE/ACM Intl. Conf. on Grid Computing, pp. 283–290. IEEE (2007)
29. Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: Analyzing market-based resource allocation strategies for the computational grid. International Journal of High Performance Computing Applications 15(3), 258–281 (2006)
30. Yao, A.C.-C.: How to generate and exchange secrets. In: 27th Annual Symp. on Foundations of Computer Science (FOCS), pp. 162–167. IEEE (1986)

Author Index